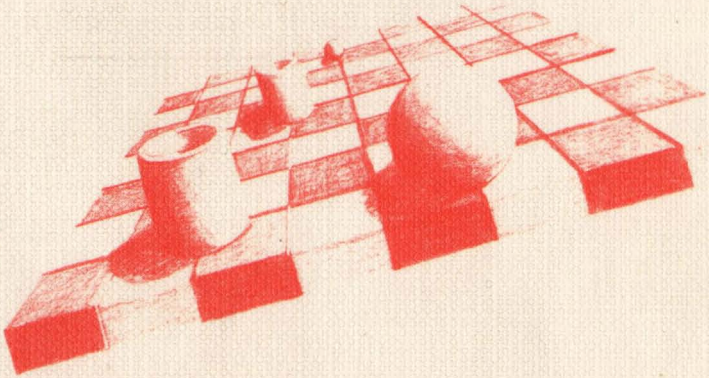
SCHNEIDER

STRATEGIESPIELE



UND WIE MAN SIE AUF DEM

ATARI®

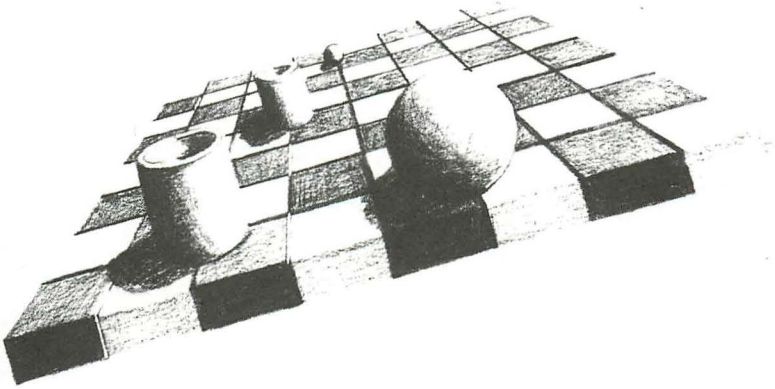
600 XL / 800 XL

PROGRAMMIERT

EIN DATA BECKER BUCH

SCHNEIDER

STRATEGIESPIELE



UND WIE MAN SIE AUF DEM

ATARI®

600 XL / 800 XL

PROGRAMMIERT

EIN DATA BECKER BUCH

ISBN 3-89011-077-0

1. Auflage

Copyright © 1985 DATA BECKER GmbH
Merowingerstraße 30
4000 Düsseldorf

Alle Rechte vorbehalten. Kein Teil dieses Buches darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne schriftliche Genehmigung der DATA BECKER GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

VORWORT

Dieses Buch ist bestimmt für all jene, die Spaß am Programmieren haben und sich gerne mit Strategiespielen beschäftigen.

Dieses Buch zeigt, wie der Computer zu einem spielstarken Gegner programmiert werden kann.

Dieses Buch enthält 8 Spielprogramme. Jedes Programm umfaßt folgende Teile:
-Bild aus dem Spielgeschehen
-Spielregeln
-Programmerläuterung
-Programmablauf
-Variablenliste
-Listing
Es wurde großen Wert darauf gelegt, die Programme klar und übersichtlich zu schreiben, so daß sie leicht verstanden werden können.
Die Programme laufen auf allen Atari Computern.

Dieses Buch dient dem Zweck, Ihr Interesse an Strategiespielen zu wecken und Ihnen die Möglichkeit zu geben, selber Strategiespiele zu programmieren. Die Spielprogramme sollen Ihnen außerdem viele unterhaltsame Stunden beim Spiel gegen den Computer bereiten.

INHALTSVERZEICHNIS

Vorwort

Erster Teil: Einfache Spiele mit feststehender Strategie

1. Einführung.....2
2. Nim mit einem Haufen.....12
3. Blockade.....21
4. Treib die Dame in die Ecke.....40

Zweiter Teil: Komplexe Spiele - Heuristische Suchverfahren

5. Bewertungsfunktion, Suchverfahren.48
6. Springer gegen Bauer.....69
7. Vier gewinnt.....143

Dritter Teil: Der Computer lernt

8. Bauernkampf.....88
9. Verbesserung der Lernmethode.....103
10. Hexapawn.....109
11. Mini-Dame.....123

Anhang

12. Literaturverzeichnis.....181
13. Liste der verwendeten Befehle....175

Erster Teil
==========
Einfache Spiele mit feststehender Strategie
==========

KAPITEL 1: EINFÜHRUNG

STRATEGIE, WAS IST DAS?

Will man ein ganz bestimmtes Ziel erreichen, so muß man sich einen genauen Plan erstellen, der bei richtigem Befolgen zu diesem Ziel führt. Solch einen Plan bezeichnen wir als Strategie. Das Ziel besteht in unserem Fall darin, ein Spiel zu gewinnen.

Beispiel für eine einfache Strategie:

Sie sind zu dick und wollen abnehmen. Mögliche Strategien wären zum Beispiel:
-weniger essen
-mehr Sport treiben
-eine Diät-Kur machen.

Sie sehen, zu dem gleichen Ziel gibt es oftmals mehrere unterschiedliche Strategien. Unter diesen Strategien sollte möglichst die Beste, d.h. die am schnellsten und am sichersten zum Ziel führende, ausgewählt werden.

Eine Strategie muß nicht immer zum Erfolg führen. Das macht sich bei unseren Computerspielen besonders bemerkbar. Für viele Spiele läßt sich keine sichere Gewinnstrategie

angeben; jedenfalls keine, die mit den begrenzten Möglichkeiten unseres Computers zu erfüllen wäre. Der begrenzte Speicherplatz und die Ausführungsgeschwindigkeit unseres Computers stecken hier schnell Grenzen. Ein Spiel übrigens, welches sich völlig analysieren läßt und bei dem der Computer eine sichere Gewinnstrategie besitzt, so daß er immer gewinnt, wird auf die Dauer natürlich langweilig. An einem solchen Spiel verliert man meist schon nach kurzer Zeit das Interesse. Wer will schon immer verlieren?

STRATEGIESPIELE

BEDEUTUNG, UNTERSCHIEDE UND PROGRAMMIERUNG

Strategiespiele sind Spiele, in denen der Zufall völlig ausgeschlossen ist. Das Spielgeschehen wird daher nur vom Geschick der Spieler bestimmt. Sämtliche Würfel- oder Kartenspiele entfallen dadurch natürlich. Bekannte und weit verbreitete Strategiespiele sind z.B. Schach, Dame, Mühle.

Die Anzahl der Spieler ist von Spiel zu Spiel verschieden, meistens ist sie auch variabel. In diesem Buch beschränken wir uns auf Spiele, die zwei Spieler erfordern. Diese zwei Spieler, also Computer und Mensch, kämpfen gegeneinander. Jeder der Spieler versucht, das Spiel für sich zu entscheiden. Die Spieler kommen dabei abwechselnd zum Zuge. In den meisten Fällen kann gewählt werden, ob der Computer oder der menschliche Spieler anfangen soll. Es kommt aber auch vor (oft zur Vereinfachung), daß genau festgelegt ist, wer anfängt.

In der Mehrzahl der Spiele sind die zwei Spieler gleichberechtigt. Jeder Spieler hat die gleichen (lediglich zur Unterscheidung andersfarbigen) Spielfiguren. Für jeden

Spieler gelten die gleichen Spielregeln. Dieses Buch zeigt aber auch Spiele, in denen die zwei Spieler völlig unterschiedliche und verschiedenen Spielregeln gehorchende Spielfiguren besitzen.

Es gibt prinzipiell zwei Möglichkeiten, einem Computer Spielstärke zu verleihen. Die erste Möglichkeit besteht darin, den Computer so zu programmieren, daß er eine konstante Spielstärke besitzt (es kann lediglich zwischen verschiedenen Schwierigkeitsstufen gewählt werden). Damit beschäftigen sich die ersten beiden Teile dieses Buches. Die zweite Möglichkeit besteht darin, den Computer während des Spieles lernen zu lassen. Diese Möglichkeit ist natürlich äußerst interessant und kann auf einem Großteil der Spiele angewandt werden. Davon, wie man einen Computer zum Lernen bringt, handelt also der dritte Teil dieses Buches.

SPIELSTÄRKE COMPUTER UND SPIELSTÄRKE MENSCH

In vielen Strategiespielen ist der Computer dem Menschen haushoch überlegen. Es gibt aber auch Spiele, bei denen sich der Computer schwertut. Ein bekanntes Beispiel hierfür ist Schach, das Paradepferd unter den Strategiespielen. Es ist bis jetzt noch keinem gelungen, einen vom Menschen unschlagbaren Schachcomputer zu entwickeln. Doch auch dies ist nur noch eine Frage der Zeit. Denn die Spielstärke der Schachcomputer hat rapide zugenommen. Noch vor wenigen Jahren wurde jeder ausgelacht, der behauptete, die Schachcomputer würden dem Menschen einmal überlegen sein. Heutzutage lacht niemand mehr darüber. Sogar die Schachprofis müssen die Spielstärke der neuen Schachcomputer anerkennen. So wurde z.B. in den

Bell-Laboratorien von Ken Thompson und Joe Candon ein "Bell" genanntes Schachprogramm entwickelt, welches eine Elozahl von 2160 besitzt (Die Elozahl gibt bekanntlich die Spielstärke eines Schachspielers an - ab 2000 darf ein Spieler den Titel "Schachmeister" führen). Mit dieser Spielstärke wird "Bell" nur noch von wenigen Großmeistern überboten. Das Programm, welches auf einem speziell für Schach konzipierten Computer läuft, überprüft im Durchschnitt ca. 160 000 Stellungen pro Sekunde.

Der Computer hat gegenüber dem Menschen übrigens einen großen Vorteil. Dieser Vorteil tritt beim Schachspiel besonders deutlich zutage. Haben Sie schon einmal Schach gespielt? Wenn ja, dann werden Sie jetzt bestimmt zustimmend nicken. Bei der Fülle der möglichen Züge und der Komplexität des Spielfeldes schleichen sich beim Schachspiel nämlich immer wieder Leichtigkeitsfehler ein. Man übersieht einfach immer wieder Züge. Bei Schach ist deshalb ein gehöriges Maß an Konzentration unerläßlich. Der Mensch unterliegt Konzentrationsschwächen, der Computer jedoch nicht. Der Computer wird deshalb auch niemals einen Leichtigkeitsfehler begehen, er übersieht keine Züge.

Schon Edgar Allan Poe hat diese Tatsache erkannt. In seiner Kurzgeschichte "Der Mord in der Rue Morgue" schreibt er über das Schachspiel:
"...Bei diesem letzten Spiel, wo die Figuren die unterschiedlichsten bizarren Bewegungen von verschiedenem und veränderlichem Wert ausführen, wird - mit einem häufigen Irrtum - etwas, das nur sehr kompliziert ist, für tief gehalten. Die Aufmerksamkeit wird stark in Anspruch genommen. Erlahmt sie einen Augenblick, so wird etwas übersehen, das zu Verlusten oder zur Niederlage führt. Bei der Mannigfaltigkeit und Unübersichtlichkeit der möglichen Züge sind die Chancen, solche Fehler zu begehen, natürlich sehr groß, und in neun von zehn Fällen wird derjenige Spieler, der sich besser konzentriert, eher gewinnen als der geistreichere."

BAUM UND GRAPH

Ein wichtiges Hilfsmittel zur Analyse von Strategiespielen ist der Baum. Was es damit auf sich hat, will ich an einem Beispiel erläutern.

Folgendes Spiel soll analysiert werden:

Auf einem 3 x 3 - Spielfeld stehen sich drei weiße und drei schwarze Bauern gegenüber. Die weißen Bauern seien durch "X", die schwarzen durch "O" gekennzeichnet.

O	O	O
X	X	X

Spieler Weiß beginnt, danach wird abwechselnd gezogen. Bei jedem Zug kann ein Bauer geradeaus vorwärts auf ein angrenzendes leeres Feld ziehen. Ist jedoch ein diagonal vorwärts liegendes angrenzendes Feld vorhanden, auf dem sich ein gegnerischer Bauer befindet, so muß dieser geschlagen werden. Bestehen mehrere Schlagmöglichkeiten, so kann eine davon ausgewählt werden. Sieger ist derjenige, welcher alle Bauern des Gegners geschlagen hat oder dessen Bauern blockiert hat, daß dieser nicht mehr ziehen kann.

Zur Analyse dieses einfachen, für unsere Zwecke jedoch bestens geeigneten Spieles fertigen wir nun den Baum an. Dazu zeichnen wir die Ausgangsstellung, auch Wurzel genannt, an den oberen Blattrand. Weiß beginnt, er hat drei Züge zur Auswahl. Nach dem ersten Zug von Weiß sind also drei verschiedene Stellungen möglich. Diese drei Stellungen

werden in einer Linie unter die Ausgangsstellung gezeichnet und mit dieser verbunden. Nun ist Schwarz am Zuge. Alle Stellungen nach dem Zug von Schwarz werden wieder in eine Linie gezeichnet und mit der hervorgehenden Stellung verbunden usw. Die einzelnen Stränge des entstehenden Diagramms werden solange fortgeführt, bis Endstellungen auftreten (alle Bauern geschlagen oder blockiert).

Der Spielablauf erfolgt in einem solchen Diagramm also von oben nach unten. Jedes mögliche Spiel ist in dem Baum enthalten. Um den Baum zu vereinfachen, werden Stellungen in einer Linie, welche zueinander symmetrisch sind (in unserem Fall also seitenverkehrt sind) nur einmal weiterentwickelt.

Die einzelnen Stellungen in dem Baum bezeichnet man als Knoten, die Verbindungslinien stellen die Züge dar und heißen Kanten. Jeder Knoten wird numeriert. Ist eine Stellung zu einer anderen symmetrisch, so wird dies unter diesem Knoten vermerkt. Endknoten enthalten einen Hinweis auf den Sieger. Außerdem wird zur besseren Übersicht am linken Blattrand die Nummer des Zuges und der am Zug befindliche Spieler aufgeführt. Das Ergebnis finden Sie auf der nächsten Seite. Machen Sie sich damit vertraut!

Haben wir den Baum gezeichnet, so macht uns die Analyse keine Schwierigkeiten mehr. Aus dem Baum erkennt man sofort, daß Weiß verliert, falls Weiß am Rand eröffnet. Eröffnet Weiß dagegen in der Mitte, so verliert Schwarz. Derjenige, welcher den ersten Zug macht, kann also immer gewinnen. Die Strategie besteht dann einfach darin, in der Mitte zu eröffnen, alles übrige ergibt sich dann von selbst.

Was aber, wenn wir das Spielfeld auf ein 3 x 4 - Spielfeld erweitern und die bisherigen Spielregeln beibehalten? Zeichnen Sie bitte zur Übung den zugehörigen Baum!

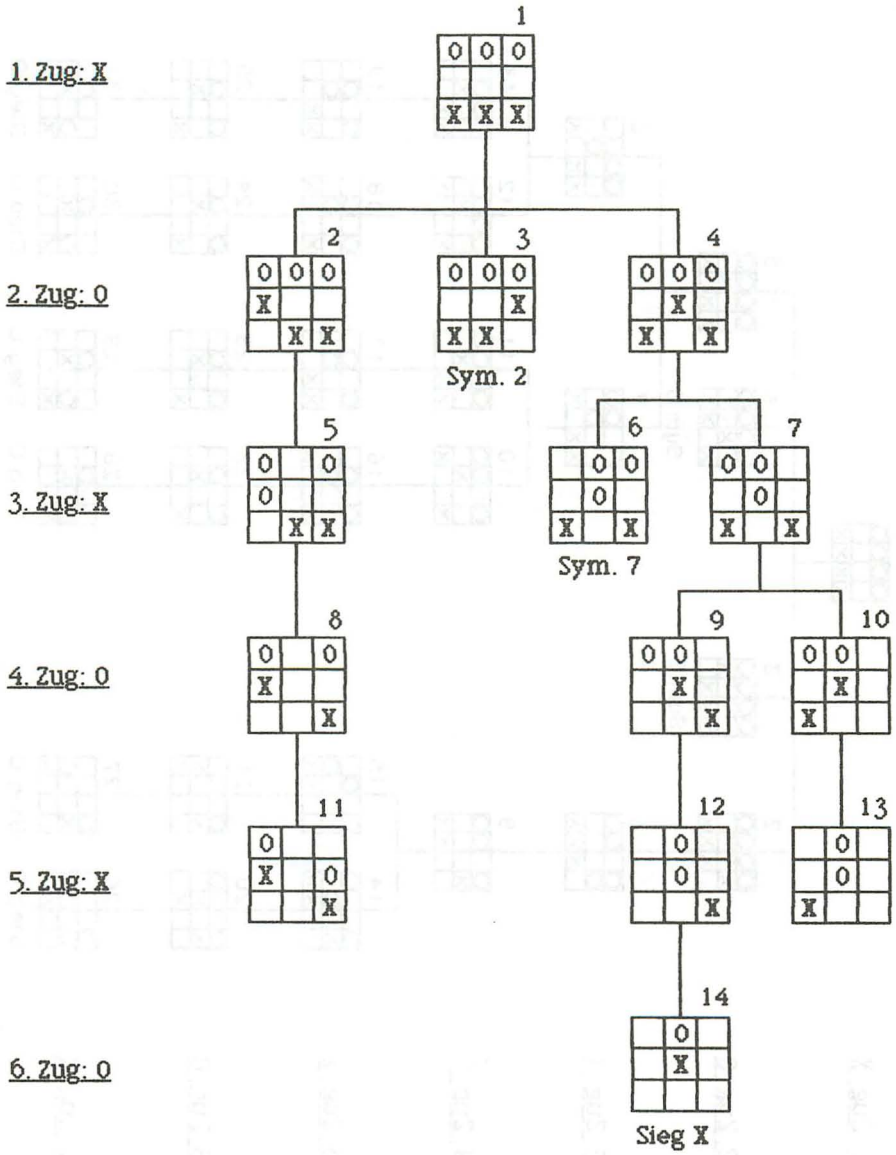
Überraschenderweise gewinnt diesmal nicht Weiß, sondern auf

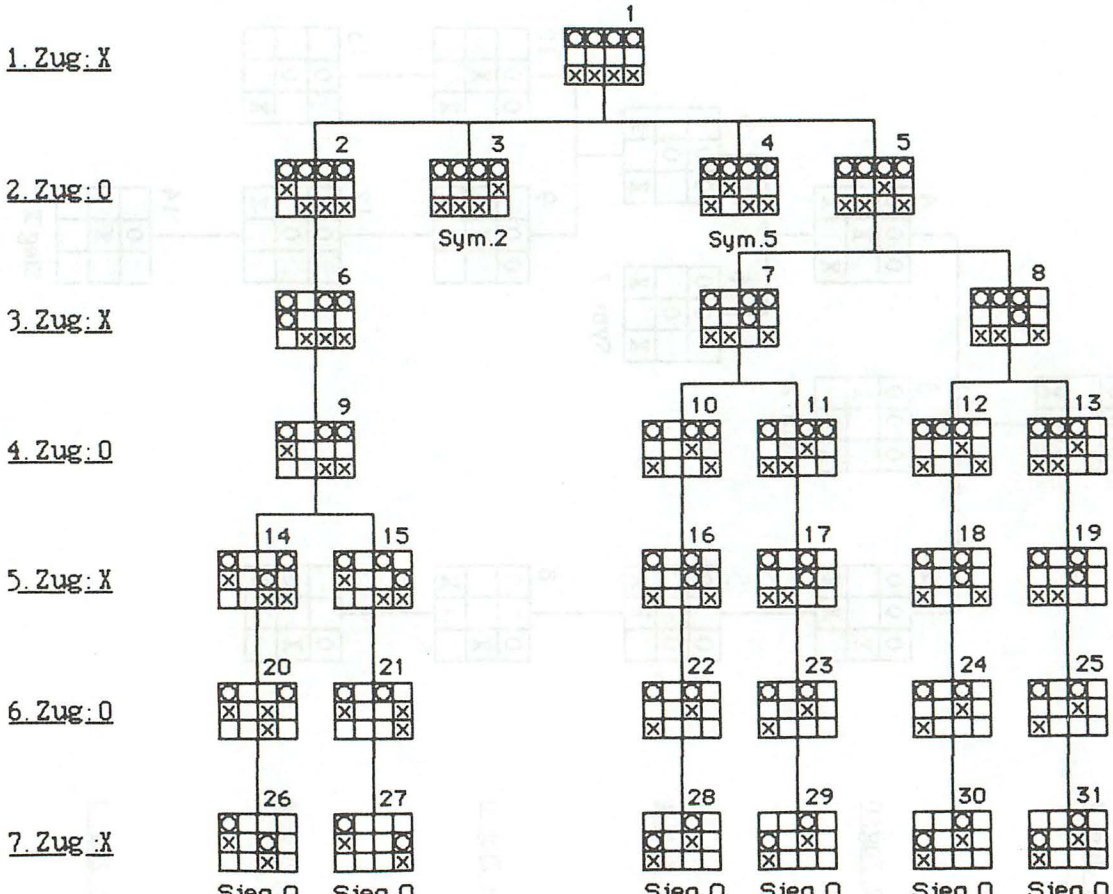
jeden Fall Schwarz. Weiß hat überhaupt keine Möglichkeit zu gewinnen, jeder Strang läuft auf einen Sieg von Schwarz hinaus.

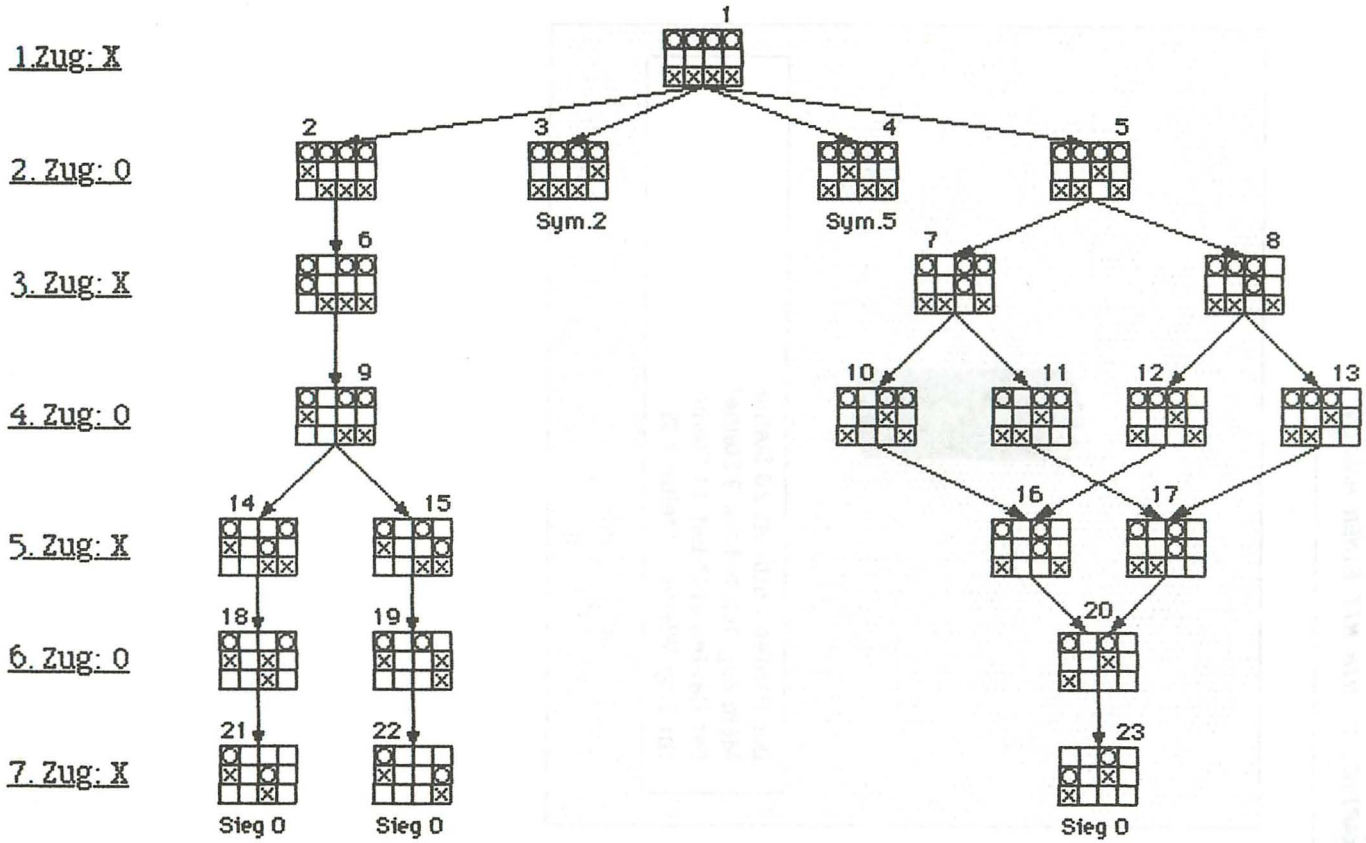
Wenn Sie den Baum betrachten, so wird Ihnen auffallen, daß die Knoten 22, 23, 24 und 25 identisch sind. Diese vier Knoten müssen in einem Baum getrennt entwickelt werden oder aber man vermerkt die Nummer des identischen Knotens, wie es im Falle der Symmetrie getan wird. Das ist deshalb so, weil in einem Baum jeder Knoten (mit Ausnahme des Startknotens) genau einen Elternknoten besitzt.

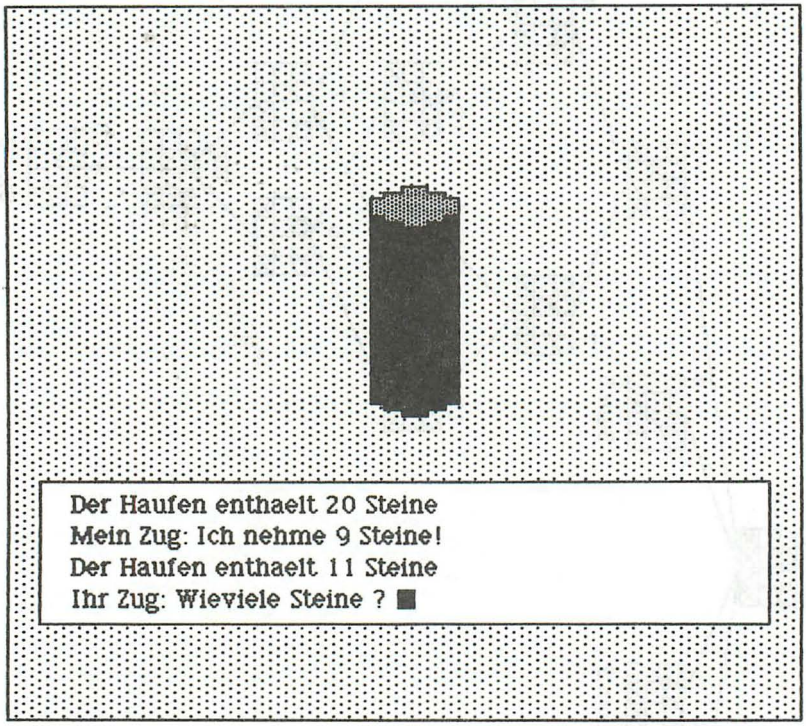
Bei größeren Diagrammen empfiehlt sich zum Zwecke der Übersicht statt dem Baum ein Graph. Ein Graph unterscheidet sich von einem Baum dadurch, daß mehrere Knoten den gleichen Elternknoten haben dürfen. Die Verbindungen zwischen den Knoten dürfen also frei gewählt werden. Die Verbindungslinien sind deshalb mit Pfeilen auszustatten. Sie heißen nun nicht mehr einfach Kanten, sondern werden nun gerichtete Kanten genannt. Ein Baum ist also nur eine spezielle Form des Graphen.

Baum


1. Zug: X
2. Zug: O
3. Zug: X
4. Zug: O
5. Zug: X
6. Zug: O
1
2
3
4
Sym. 2
5
6
7
Sym. 7
8
9
10
11
12
13
14
Sieg X


1. Zug: X
2. Zug: O
3. Zug: X
4. Zug: O
5. Zug: X
6. Zug: O
7. Zug: X
1
2
3
Sym.2
4
Sym.5
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
Sieg O
Sieg O
Sieg O
Sieg O
Sieg O
Sieg O


1.Zug: X
2. Zug: 0
3. Zug: X
4. Zug: 0
5. Zug: X
6. Zug: 0
7. Zug: X
1
2
3
4
5
Sym.2
Sym.5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Sieg 0
Sieg 0
Sieg 0


Der Haufen enthaelt 20 Steine
Mein Zug: Ich nehme 9 Steine!
Der Haufen enthaelt 11 Steine
Ihr Zug: Wieviele Steine ?

SPIELREGELN, STRATEGIE

Eine Gewinnstrategie läßt sich bei vielen Spielen dadurch finden, daß man nicht von der Anfangsstellung sondern von der Endstellung ausgeht. Folgende einfache Version des bekannten NIM-Spieles macht dies deutlich:

Es sei ein Haufen von 10 Steinen gegeben. Spieler A beginnt, danach wechseln sich Spieler A und Spieler B ab. Bei jedem Zug darf ein Spieler einen, zwei oder drei Steine wegnehmen. Derjenige Spieler, welcher den letzten Stein nimmt, hat gewonnen.

Die Stellungen sind durch eine einzige Zahl gekennzeichnet, welche die Anzahl der Steine angibt. Da ein Spieler maximal drei Steine wegnehmen kann, waren vor dem letzten Zug folglich noch 1,2 oder 3 Steine vorhanden. Derjenige Spieler, welcher nach seinem Zug 1,2 oder 3 Steine übrigläßt, hat verloren. Die Strategie besteht also darin, den Gegner zu zwingen, 1,2 oder 3 Steine übrigzulassen. Welche Stellung erfüllt diese Bedingung? Nun, Sie haben es sicher schon bemerkt, es ist die Stellung mit 4 Steinen. Derjenige Spieler, welcher vor seinem Zug vier Steine vorfindet, hat verloren, da nach seinem Zug 1,2 oder 3 Steine übrigbleiben, die sein Gegner alle wegnehmen kann. Gehen wir nun auf dieselbe Weise weiter. Vier Steine ist unser Ziel. Vier Steine können erreicht werden, wenn vor dem Zug 5,6 oder 7 Steine vorhanden sind. Lassen wir nach dem Zug 8 Steine liegen, so können wir nach unserem nächsten Zug 4 Steine liegenlassen und gewinnen folglich.

Die Strategie haben wir rekursiv erhalten. Sie ist recht einfach und besteht darin, den Zug so zu wählen, daß acht oder vier Steine übrigbleiben.

NIM MIT EINEM HAUFEN - ALLGEMEINE FORMULIERUNG

Wir wollen nun dieses einfache NIM-Spiel etwas allgemeiner formulieren. Wieder haben wir einen Haufen mit Steinen. Die Anzahl der Steine sei jedoch beliebig, wir bezeichnen sie mit A. Außerdem wird von den Spielern vereinbart, wieviele Steine maximal weggenommen werden dürfen. Die Anzahl der wegzunehmenden Steine darf also eine vereinbarte Zahl B nicht überschreiten.

Im vorigen Beispiel hatten wir A=10, B=3. Vier Steine und acht Steine waren die beiden Gewinnstellungen.

Gehen wir wieder von der Endstellung aus. Da maximal B Steine weggenommen werden dürfen, müssen wir nach unserem Zug B+1 Steine liegenlassen. B+1 Steine ist die erste Gewinnstellung. B+1 Steine erhalten wir, wenn wir nach dem Zug 2(B+1) Steine übriglassen usw.

Die Strategie lautet allgemein formuliert nun also:
Der Zug ist so zu wählen, daß die Anzahl der restlichen Steine bei der Division durch B+1 den Rest Null läßt, oder anders formuliert, die Anzahl der übrig bleibenden Steine muß ein Vielfaches von B+1 sein.

Zu diesem Spiel wird Ihnen auf den nächsten Seiten ein Programm vorgestellt und erläutert.

PROGRAMMERLÄUTERUNG

Zunächst werden im Grafikmodus Null die Spielregeln ausgegeben, danach erfolgt in den Zeilen 1200 bis 1280 die Eingabe der notwendigen Daten. Eingegeben werden muß vom Spieler die Anzahl A der Steine auf dem Haufen, die Anzahl B der wegzunehmenden Steine und die Variable H, mit der der Spieler wählen kann, ob er oder der Computer anfangen soll.

Nun wird der Bildschirm gesetzt. Dazu wird Grafikmodus 7 mit vier Zeilen Text aufgerufen. Dann erfolgt die Ausgabe des Haufens mit A Steinen auf dem Bildschirm.

Anschließend beginnt das Hauptprogramm. Es gibt die Anzahl der verbliebenen Steine aus und ruft den Spielerzug bzw. den Computerzug auf. Ist nach dem Zug die Variable A größer Null, so sind noch Steine übrig. Der Computer ändert dann in Zeile 3150 die Variable H und springt wieder zurück nach Zeile 3110. Ist die Variable A gleich Null, so sind alle Steine weggenommen, das Spiel ist also zu Ende. In diesem Fall wird der Gewinner ausgegeben und der Spieler kann wählen, ob er ein neues Spiel beginnen will.

Der Spielerzug beginnt in Zeile 4000. Nach dem Eingeben der Variablen S wird in Zeile 4030 getestet, ob die Eingabe korrekt ist. Dies ist nicht der Fall, wenn S kleiner als 1 ist, da mindestens ein Stein genommen werden muß. Außerdem darf S nicht größer als A und B sein. War die Eingabe korrekt, so erfolgt ein Return ins Hauptprogramm.

Der Computerzug liegt in den Zeilen 5000 bis 5040. Interessant ist Zeile 5010. Um die Anzahl der wegzunehmenden Steine zu ermitteln, könnte man eine Schleife von 1 bis B durchlaufen und jedesmal testen, ob

die Anzahl der Steine minus dieser Schleifenvariablen durch B+1 teilbar ist. Diese Methode wäre jedoch äußerst umständlich. Besser ist es, den Abstand S bis zur nächsten, durch B+1 teilbaren Zahl, direkt zu berechnen. Zur Erinnerung, Int(X) ermittelt die größte ganze Zahl, die kleiner oder gleich X ist. Im Falle S=0 ist dem Computer der Spieler zuvorgekommen. Sollte deshalb S gleich Null sein, macht der Computer einen Verlegenheitszug und nimmt einen Stein.
Nach dem Spieler- bzw. Computerzug muß der auf dem Bildschirm dargestellte Haufen um die Anzahl der weggenommenen Steine verkleinert werden. Dies geschieht in den Zeilen 6000 bis 6110. Das Wegnehmen eines Steines geschieht dadurch, daß der obere Teil des Haufens einfach ein Stück nach unten geschoben wird. Dadurch verkleinert sich der Haufen von oben nach unten.
Wer die einfache Gewinnstrategie nicht kennt, wird gegen den Computer keine Chance haben und von der Spielstärke desselben in höchstem Maße erstaunt sein.

PROGRAMMABLAUF

Nim mit einem Haufen

Sie müssen eingeben, wieviele Steine sich auf dem Haufen befinden und wieviele Steine maximal weggenommen werden dürfen. Der Spieler, welcher den letzten Stein nimmt, hat das Spiel gewonnen!

```
Wieviele Steine auf Haufen (3-20)? 10
Wieviele Steine maximal wegnehmen? 3
Wollen Sie anfangen (J/N)? N

Der Haufen enthält 10 Steine.
Mein Zug: Ich nehme 2 Steine!
Der Haufen enthält 8 Steine.
```

```
Ihr Zug: Wieviele Steine? 2
Der Haufen enthält 6 Steine.
Mein Zug: Ich nehme 2 Steine!
Der Haufen enthält 4 Steine.
Ihr Zug: Wieviele Steine? 1
Der Haufen enthält 3 Steine.
Mein Zug: Ich nehme 3 Steine!

Ich habe gewonnen!!

Noch ein Spiel (J/N)? N
Schade
```

VARIABLENLISTE

A$	:Eingabestring der Länge 1
A	:Anzahl der Steine auf dem Haufen
B	:Anzahl maximal wegnehmbarer Steine
H	:Indikator für am Zug befindlichen Spieler
ANF	:Anfangsadresse des Bildschirmspeichers Grafikmodus 7, d.h. die Adresse der linken oberen Ecke des Bildschirms.
S	:Anzahl der Steine, die Spieler oder Computer wegnimmt.

Listing: Nim mit einem Haufen

```
1000 Rem +++ Anleitung ++++++++++++++++++++++++++++++++++++++++
1010 Graphics 0: Setcolor 2,12,0: Setcolor 4,2,6
1020 Dim A$(1)

1100 Rem --- Spielregeln
1110 Print "Nim mit einem Haufen"
1120 Print
1130 Print "Sie muessen eingeben, wieviele Steine"
1140 Print "sich auf dem Haufen befinden und wie-"
1150 Print "viele Steine maximal weggenommen"
1160 Print "werden duerfen. Der Spieler, welcher"
1170 Print "den letzten Stein nimmt, hat das"
1180 Print "Spiel gewonnen!"

1200 Rem --- Eingabe der Daten
1210 Print
1220 Print "Wieviele Steine auf Haufen (3-20) ";
1230 Input A: If A<3 or A>20 then 1220
1240 Print "Wieviele Steine maximal wegnehmen ";
1250 Input B: If B<2 or B>A then 1240
1260 Print "Wollen Sie anfangen (J/N) ";
1270 Input A$
1280 H=1: If A$="N" then H=2

2000 Rem +++ Bildschirm setzen
2010 Graphics 7: Setcolor 0,2,4
2020 X=70: Y=74
2030 For I=1 to A
2040 Sound 0,250-10*I,30,10
2050 For J=0 to 2: For K=0 to 2
2060 Color 1: X1=X+3*J+K: Y1=Y+J: Plot X1,Y1: Drawto X1,Y1+2:
     Plot 157-X1,Y1: Drawto 157-X1,Y1+2
2070 Color 2: Y2=Y-3-J: Plot X1,Y2: Drawto X1,Y1-1: Plot 157-
     X1,Y2: Drawto 157-X1,Y1-1
2080 Next K: Next J
2090 Y=Y-3: Next I
2100 Sound 0,0,0,0
2110 ANF=Peek(89)*256+Peek(88)
```

```
3000 Rem +++ Hauptprogramm ++++++++++++++++++++++++++++++++++++

3100 Rem --- Aufruf der Unterprogramme
3110 Print "Der Haufen enthaelt ";A;" Steine"
3120 If H=1 then gosub 4000  :Rem Spielerzug
3130 If H=2 then gosub 5000  :Rem Computerzug
3140 Gosub 6000              :Rem Steine wegnehmen
3150 If A>0 then H=3-H: Goto 3110

3200 Rem --- Spielende
3210 Setcolor 2,5,4
3220 Print
3230 For I=15 to 0 Step -2: For J=50 to 30 Step -1: Sound 0,J,
     10,I: Next J: Next I: Sound 0,0,0,0
3240 If H=1 then print "Sie haben gewonnen!!"
3250 If H=2 then print "Ich habe gewonnen!!"
3260 Print
3270 Print "Noch ein Spiel (J/N) ";
3280 Input A$
3290 If A$="J" then Run
3300 Print "Schade": End

4000 Rem +++ Spielerzug +++++++++++++++++++++++++++++++++++++++
4010 Print "Ihr Zug: Wieviele Steine ";
4020 Input S
4030 If S<1 or S>A or S>B then 4010
4040 Return

5000 Rem +++ Computerzug ++++++++++++++++++++++++++++++++++++++
5010 S=A-Int(A/(B+1))*(B+1)
5020 If S=0 then S=1
5030 Print "Mein Zug: Ich nehme ";S;" Steine!"
5040 Return

6000 Rem +++ S Steine wegnehmen ++++++++++++++++++++++++++++++
6010 For I=A to A-S+1 Step -1
6020 Sound 0,250-10*I,30,10
6030 For J=0 to 9
6040 For K=ANF+3137-40*(I-1)*3-J*40 to ANF+3142-40*(I-1)*3-J*40
6050 Poke K,Peek(K-3*40)
6060 If I=1 then Poke K,0
6070 Next K
```

```
6080 Next J
6090 Next I
6100 Sound 0,0,0,0: A=A-S
6110 Return
```

KAPITEL 3: BLOCKADE

```
Blockade

Wollen Sie anfangen (J/N) ?J

C C C C C C C C
S S S S S S S S
1 2 3 4 5 6 7 8

Welche Spalte ?
```

Nehmen wir an, Spieler Weiß beginnt. Er kann jeden seiner Bauern um 1 bis 6 Felder vorrücken, für jeden seiner Bauern stehen ihm also sechs Möglichkeiten zur Auswahl. Bei acht Bauern macht das insgesamt 48 Zugmöglichkeiten. Einen Graphen zu zeichnen hat damit natürlich wenig Sinn.

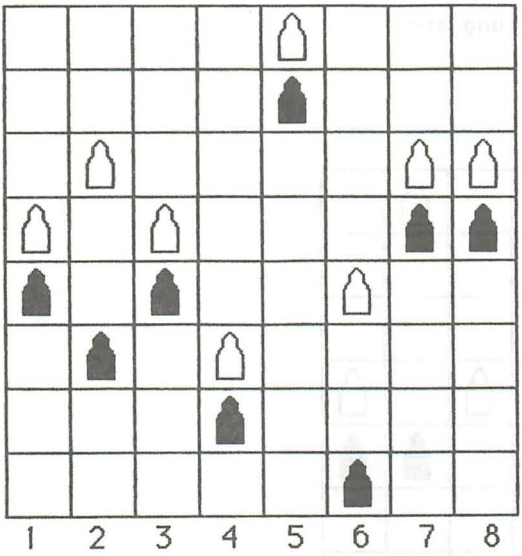
Versuchen Sie erst einmal selbst, eine Strategie zu entwickeln! Auf den folgenden Seiten wird Ihnen eine Strategie vorgestellt.

SPIELANALYSE

Zunächst einmal ist festzustellen, daß Blockade im Prinzip nichts anderes als ein geschickt getarntes Nim-Spiel ist! Blockade entspricht dem Nim-Spiel mit acht Haufen zu je sechs Steinen, wobei die Anzahl der in einem Zug wegnehmbaren Steine eines Haufens nicht begrenzt ist. Die acht Spalten entsprechen den acht Haufen, die sechs leeren Felder jeder Spalte entsprechen den sechs Steinen jedes Haufens. Rückt ein Bauer in einer Spalte ein oder mehrere Felder vor, so ist das dasselbe, als wenn vom entsprechenden Haufen die gleiche Anzahl Steine wie vorgerückte Felder weggenommen wird. Blockieren sich zwei gegnerische Bauern, so heißt das, daß alle Steine des Haufens weggenommen worden sind. Machen Sie sich diesen Zusammenhang klar! Zwei Spiele, die auf den ersten Blick völlig unterschiedlich sind, erweisen sich bei genauerem Betrachten nicht selten als gleich, so daß für beide Spiele dieselbe Strategie gilt.

Hat man überhaupt keine Idee, so ist es am besten, man spielt ein paar Partien. Verliert man oder gewinnt man eine Partie, so muß man sich fragen, ***WARUM*** man diese Partie verloren bzw. gewonnen hat. Das wollen wir nun tun. Nehmen

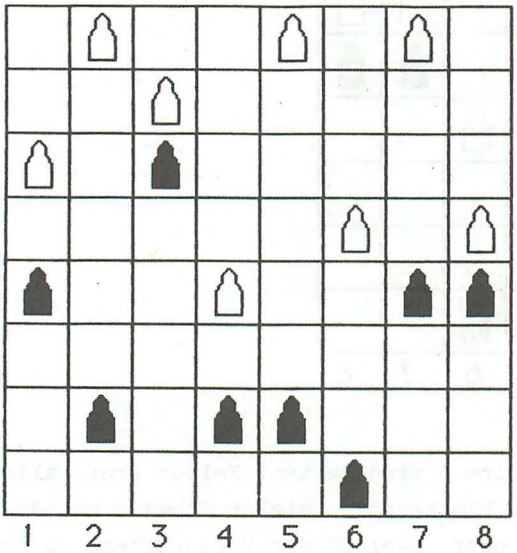
wir also an, daß Spieler Weiß und Spieler Schwarz im Laufe einer Partie zu folgender Stellung gelangen, Schwarz sei am Zuge:



In Spalte 2 und Spalte 6 sind je zwei Felder frei, alle übrigen Spalten sind blockiert. Spieler Schwarz hat die Möglichkeit, um ein oder zwei Felder vorzurücken. Rückt Schwarz um zwei Felder vor, so hat er verloren, wenn Weiß ebenfalls zwei Felder vorrückt. Rückt Schwarz um ein Feld vor, so hat er aber ebenfalls verloren, wenn Weiß in der anderen Spalte um ein Feld vorrückt, da dann in Spalte 2 und 6 je ein Feld frei ist. Diese Stellung ist also eine Verluststellung für den am Zug befindlichen Spieler. Spieler Schwarz verliert, da die beiden nicht blockierten Spalten dieselbe Anzahl freier Felder aufweisen. Macht nämlich Spieler Schwarz einen Zug, so braucht Weiß nur um die gleiche Anzahl Felder in der anderen Spalte vorrücken. Auf diese Weise hat Weiß immer noch einen Zug nach dem Zug von Schwarz.

Mit dieser Erkenntnis sind wir der endgültigen Strategie auf der Spur. Alle Stellungen, in denen noch genau zwei Spalten nicht blockiert sind, die die gleiche Anzahl freier Felder aufweisen, sind also Teilziele.

Bauen wir nun diesen ersten Ansatz weiter aus. Schauen wir uns dazu folgende Stellung an:

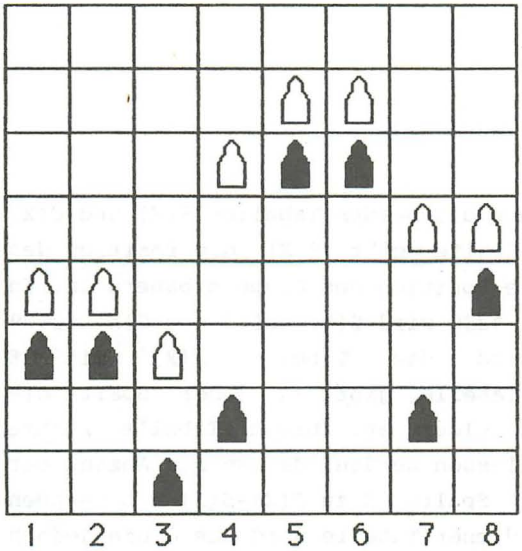


Wir haben zweimal Null freie Felder, zweimal ein freies Feld, zweimal drei freie Felder und zweimal fünf freie Felder. Mit den gleichen Überlegungen wie vorher gelangen wir zu dem Schluß, daß diese Stellung für den am Zug befindlichen Spieler eine Verluststellung ist.

Damit haben wir nun eine einfache Strategie. Teilziele sind all jene Stellungen, bei denen jeweils zwei Spalten die

gleiche Anzahl freier Felder aufweisen. Die Ausgangsstellung ist so eine Stellung. Der Spieler, welcher den ersten Zug macht, verliert also immer, sofern sein Gegenüber die Strategie kennt!

Damit sind wir jedoch noch nicht ganz fertig. Folgende einfache Stellung mag dies verdeutlichen:



Diesmal haben wir einmal ein freies Feld, einmal zwei freie Felder und einmal drei freie Felder. Spieler Schwarz sei am Zuge. Spielen Sie einmal alle Möglichkeiten von Schwarz durch. Sie werden dann bemerken, daß Schwarz das Spiel auf jeden Fall verlieren wird.
Diese Stellung ist also ein Teilziel! Wir müssen die Definition der Teilziele derart erweitern, daß damit alle Teilziele erfaßt sind. Womit ist aber obige Stellung gekennzeichnet? Sie fällt völlig aus dem bisherigen Rahmen.

Ich will die exakte Beschreibung der Teilziele nun ohne weiteren Beweis angeben. Die Beschreibung stammt von Bouton:

Man zerlege die Anzahl der freien Felder jeder Spalte in Zweierpotenzen. Genau dann ist eine Stellung ein Teilziel, wenn die Anzahl jeder Zweierpotenz gerade ist!
In unserem Beispiel erhalten wir 1=2^0, 2=2^1, 3=2^1+2^0. Die Zweierpotenzen 2^1 und 2^0 treten also jeweils genau zweimal auf. Deshalb ist diese Stellung ein Teilziel.

PROGRAMMERLÄUTERUNG

Für das Spielfeld werden die beiden Tabellen S(X) und C(X) verwendet. Zu jeder Spalte gibt S(X) die Position des Spielerbauern, C(X) die Position des Computerbauern an. In den Zeilen 1210 bis 1230 wird S(X) auf 1 und C(X) auf 8 gesetzt. Außerdem wird die Tabelle A(X) mit 6 initialisiert. Diese Tabelle gibt zu jeder Spalte die Anzahl der freien Felder an. Diese Tabelle könnte eigentlich auch weggelassen werden, da man die Anzahl der freien Felder in der Spalte X zu C(X)-S(X)-1 berechnen kann. Durch Benutzen dieser Tabelle wird das Ganze jedoch etwas übersichtlicher.

In der Tabelle P(X) werden die Zweierpotenzen festgehalten. Es ist P(1)=1=2^0, P(2)=2=2^1 und P(3)=4=2^2. Diese Tabelle ist notwendig, da die in Basic vorhandene Potenzfunktion aufgrund der Rechenungenauigkeit nicht exakte Werte liefert. Diese Tabelle P(X) wird in Zeile 1240 initialisiert.

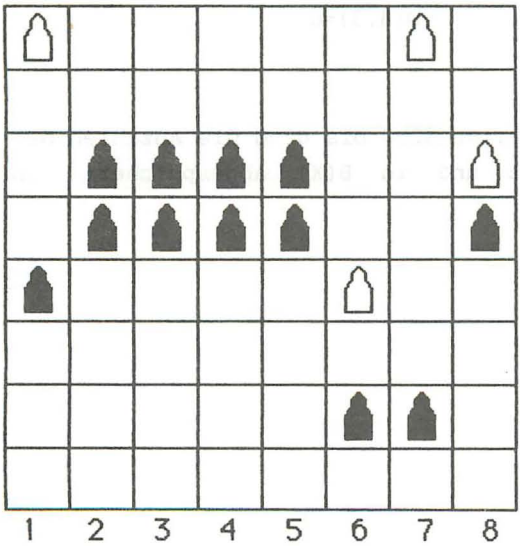
In Zeile 2150 muß der Spieler wie gewohnt eingeben, wer anfangen soll. Die Variable h dient wieder als Indikator für den am Zug befindlichen Spieler.

In den Zeilen 2200 bis 2250 werden die einzelnen Unterprogramme aufgerufen. Außerdem wird nach dem Spielerzug bzw. nach dem Computerzug der Inhalt der Tabelle A(X) zu der Variablen A addiert. Ist danach A=0, so sind alle Spalten blockiert. Das Spiel ist damit zu Ende und der Computer springt in Zeile 2240 zum "Spielende".

Der "Spielerzug" beginnt in Zeile 5000 und endet in Zeile 5320. Der Spieler muß die gewünschte Spalte S und die Anzahl F der vorzurückenden Felder eingeben. In Zeile 5310 wird der Zug ausgeführt, danach erfolgt ein Rücksprung ins "Hauptprogramm".

Der "Computerzug" (Zeile 6000) bildet den Kern des Programmes. Zuerst wird die Anzahl der freien Felder jeder Spalte in Zweierpotenzen zerlegt. Dies geschieht in den Zeilen 6100 bis 6160. Die Tabelle D(X,Y) dient zur Speicherung der gewonnenen Werte. Es ist A(X)=1*d(x,1)+2*D(X,2)+4*D(X,3), wobei D(X,Y) nur Nullen und Einsen enthält. Für Interessierte: Von A(X) wird die Dualdarstellung gebildet.

Zum besseren Verständnis wollen wir das Vorgehen des Computers an einem Beispiel verfolgen. Der Computer habe dazu folgende Stellung vor sich:



In Spalte 1 sind drei Felder frei, in Spalte 6 ist ein Feld frei und in Spalte 7 sind fünf Felder frei. Es ist also:

A(1)=3
A(2)=0
A(3)=0
A(4)=0
A(5)=0
A(6)=1
A(7)=5
A(8)=0

Für die Tabelle D(X,Y) erhalten wir:

D(1,1)=1	D(1,2)=1	D(1,3)=0
D(2,1)=0	D(2,2)=0	D(2,3)=0
D(3,1)=0	D(3,2)=0	D(3,3)=0
D(4,1)=0	D(4,2)=0	D(4,3)=0
D(5,1)=0	D(5,2)=0	D(5,3)=0
D(6,1)=1	D(6,2)=0	D(6,3)=0
D(7,1)=1	D(7,2)=0	D(7,3)=1
D(8,1)=0	D(8,2)=0	D(8,3)=0

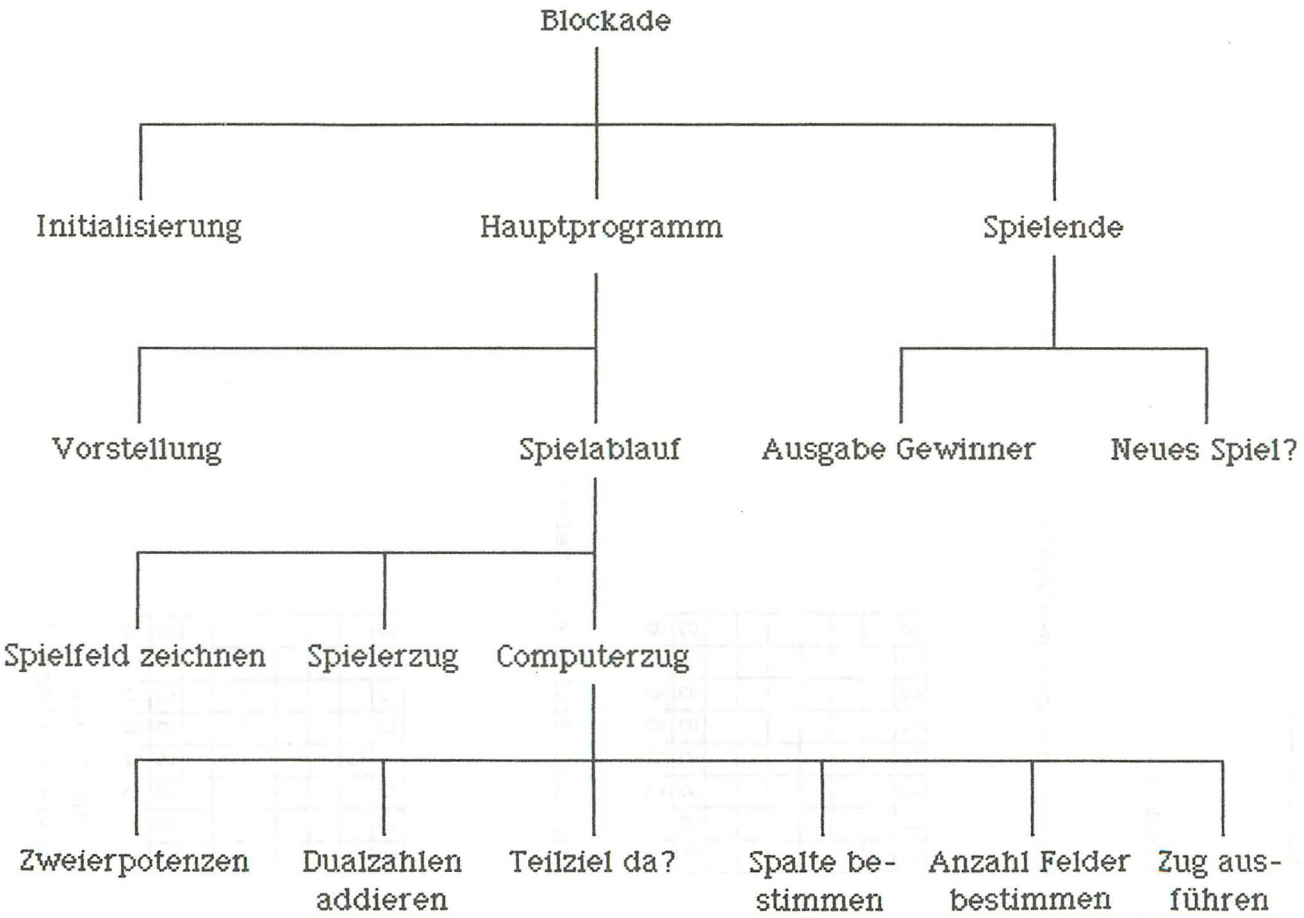
Nun werden in den Zeilen 6200 bis 6260 die Anzahlen der Zweierpotenzen gebildet und in B(X) abgespeichert. In unserem Falle ist:

B(1)=3
B(2)=1
B(3)=1

Ein Teilziel ist ja dadurch gekennzeichnet, daß die Anzahl jeder Zweierpotenz gerade ist. Der Computer testet nun, ob dies schon der Fall ist. Wenn ja, so macht der Computer einen Verlegenheitszug, indem er in einer zufällig ausgewählten Spalte um ein Feld vorrückt. Ansonsten springt der Computer in Zeile 6320 nach Zeile 6400. Der Computer kann durch einen bestimmten Zug ein Teilziel erreichen. Es gilt nun, diesen Zug zu ermitteln. Eine Möglichkeit bestünde darin, alle Züge des Computers auszuführen und nach jedem Zug zu testen, ob ein Teilziel vorliegt. Diese Möglichkeit wäre äußerst umständlich und würde viel Zeit in Anspruch nehmen. Es ist besser, den Zug direkt zu bestimmen.

In den Zeilen 6400 bis 6470 wird zuerst die Spalte bestimmt, in der der Bauer des Computers vorrücken soll. Dazu wird die höchste Zweierpotenz bestimmt, die eine ungerade Anzahl mal vorkommt. Das wäre also die 4 und es wird J gleich 3. Die Spalte S ist nun einfach diejenige Spalte mit der größten Anzahl freier Felder, bei der die Zweierpotenz J enthalten ist. In unserem Beispiel erhalten wir dadurch S gleich 7.
Nun muß noch die Anzahl F der vorzurückenden Felder bestimmt werden. F muß so gewählt werden, daß jede Zweierpotenz paarweise auftritt. Wir wissen, die Zweierpotenz J kommt ungerade mal vor, es wird deshalb F=P(J) gesetzt. Ist J=1, so sind wir fertig. Ansonsten jedoch müssen die Zweierpotenzen kleiner J noch überprüft werden. Dies geschieht nun in den Zeilen 6530 bis 6550. Trifft der Computer auf eine Zweierpotenz, die nicht gerade einmal vorkommt, so geschieht folgendes: Ist die Zweierpotenz in der Spalte S enthalten, so wird F um diese Zweierpotenz erhöht. Ansonsten wird diese Zweierpotenz von F abgezogen. Auf diese Weise erhält man eine Stellung, bei der alle Zweierpotenzen paarweise auftreten.

In unserem Beispiel wird F in Zeile 6510 auf 4 gesetzt. B(2) ist 1, also wird F um 2 erniedrigt. B(1) ist 3, es wird dadurch F um 1 erhöht. Der Computer erhält also F=3. Der Computer rückt also in Spalte 7 um drei Felder vor.


Blockade
Initialisierung
Hauptprogramm
Spielende
Vorstellung
Spielablauf
Ausgabe Gewinner
Neues Spiel?
Spielfeld zeichnen
Spielerzug
Computerzug
Zweierpotenzen
Dualzahlen addieren
Teilziel da?
Spalte bestimmen
Anzahl Felder bestimmen
Zug ausführen

PROGRAMMLAUF

Blockade

Wollen Sie anfangen(J/N)? N

C	C	C	C	C	C	C	C
S	S	S	S	S	S	S	S
1	2	3	4	5	6	7	8

Mein Zug: Spalte 4, 1 Felder

C	C	C		C	C	C	C
			C				
S	S	S	S	S	S	S	S
1	2	3	4	5	6	7	8

Welche Spalte? 7
Wieviele Felder? 5

C	C	C		C	C	C	C
			C				
						S	
S	S	S	S	S	S		S
1	2	3	4	5	6	7	8

Mein Zug: Spalte 1, 4 Felder

C	C	C		C	C	C	C
			C				
						S	
C							
S	S	S	S	S	S		S
1	2	3	4	5	6	7	8

Welche Spalte? 3
Wieviele Felder? 2

usw.

VARIABLENLISTE

S(X)	:Positionen der Bauern des Spielers
C(X)	:Positionen der Bauern des Computers
A(X)	:Anzahl freier Felder einer Spalte zwischen dem Bauern des Computers und des Spielers
D(X,Y)	:Die Anzahl der freien Felder einer Spalte ist hier dual festgehalten
B(X)	:Gibt nach dualer Zerlegung der freien Felder die Anzahl der jeweiligen Zweierpotenz an
P(X)	:Speicherung der Zweierpotenzen 2^0, 2^1 2^2
A$	:Eingabestring der Länge 1
H	:Zeiger für am Zug befindlicher Spieler
S	:Gibt die Spalte an
F	:Gibt die Anzahl der vorzurückenden Felder an

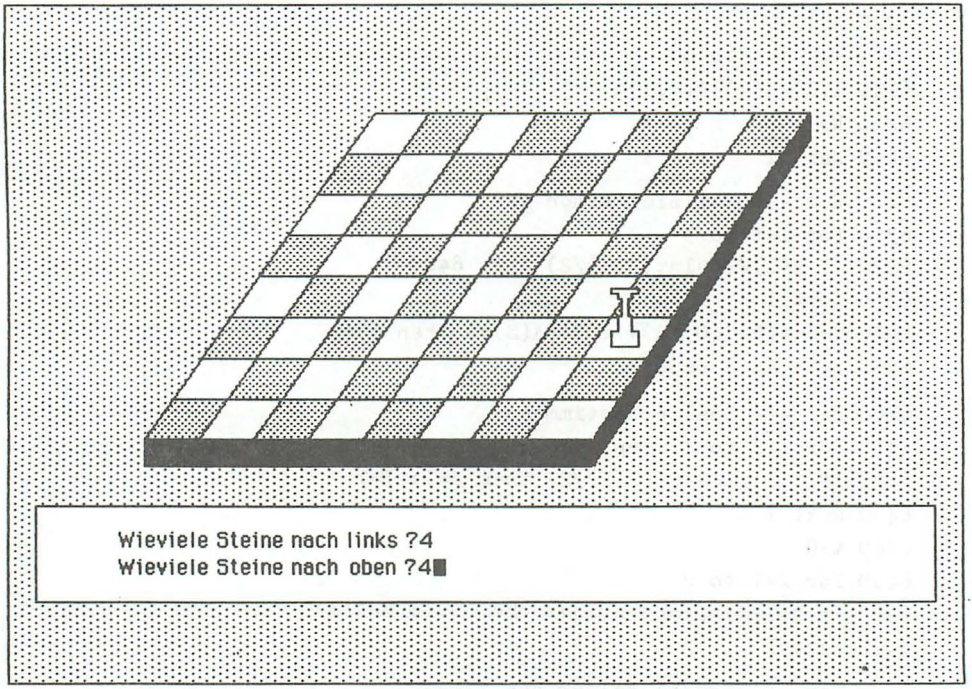
Listing: Blockade

```
1000 Rem +++ Initialisierung +++++++++++++++++++++++++++++++++++

1100 Rem --- Dimensionierung
1110 Dim S(8),C(8),A(8)          :Rem Spielfeld
1120 Dim D(8,3)                  :Rem Dualdarstellung
1130 Dim B(3),P(3)               :Rem Zweierpotenzen
1140 Dim A$(1)                   :Rem Eingabestring

1200 Rem --- Setzen der Felder
1210 For I=1 to 8
1220 S(I)=1: C(I)=8: A(I)=6
1230 Next I
1240 P(1)=1: P(2)=2: P(3)=4

2000 Rem +++ Hauptprogramm +++++++++++++++++++++++++++++++++++

2100 Rem --- Vorstellung
2110 Graphics 0: Setcolor 2,12,0: Setcolor 4,3,4
2120 Print "Blockade"
2130 Print
2140 Print "Wollen Sie anfangen (J/N) ";
2150 Input A$
2160 H=1: If A$="N" then H=2

2200 Rem --- Spielablauf
2210 Gosub 4000                  :Rem Spielfeld zeichnen
2220 If H=1 then gosub 5000      :Rem Spielerzug
2230 If H=2 then gosub 6000      :Rem Computerzug
2240 A=0: For I=1 to 8: A=A+A(I): Next I: If A=0 then 3000
2250 H=3-H: Goto 2210

3000 Rem +++ Spielende ++++++++++++++++++++++++++++++++++++++++

3100 Rem --- Ausgabe des Gewinners
3110 Print
3120 If H=1 then print "Sie haben leider gewonnen!"
3130 If H=2 then print "Ich habe gewonnen!"
```

```
3200 Rem --- Neues Spiel?
3210 Print
3220 Print "Wollen Sie noch ein Spiel (J/N) ";
3230 Input A$
3240 If A$="J" then Run
3250 End

4000 Rem +++ Spielfeld zeichnen ++++++++++++++++++++++++++++++++
4010 Print
4020 Print "┌┬┬┬┬┬┬┬┬┐"
4030 For I=8 to 1 step -1
4040 For J=1 to 8
4050 If S(J)=I then print "│S";: Goto 4080
4060 If C(J)=I then print "│C";: Goto 4080
4070 Print "│ ";
4080 Next J
4090 Print "│"
4100 Print "├┼┼┼┼┼┼┼┼┤"
4110 Next I
4120 Print " 1 2 3 4 5 6 7 8"
4130 Print: Return

5000 Rem +++ Spielerzug ++++++++++++++++++++++++++++++++++++++++

5100 Rem --- Eingabe des Zuges
5110 Print "Welche Spalte ";
5120 Input S: If S<1 or S>8 then 5110
5130 Print "Wieviele Felder ";
5140 Input F: If F<1 or F>6 then 5130

5200 Rem --- Zugkontrolle
5210 If F>A(S) then print "Zug unkorrekt!": Goto 5100

5300 Rem --- Ausfuehren des Zuges
5310 S(S)=S(S)+F: A(S)=A(S)-F
5320 Return

6000 Rem +++ Computerzug ++++++++++++++++++++++++++++++++++++++++

6100 Rem --- Zerlegung in Zweierpotenzen
6110 For I=1 to 8
6120 D=A(I)
6130 For J=1 to 3
```

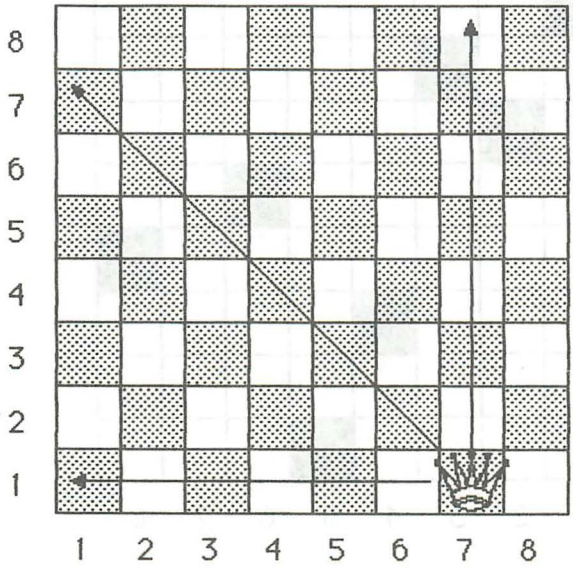
```
6140 D=D/2: D(I,J)=2*(D-Int(D)): D=Int(D)
6150 Next J
6160 Next I

6200 Rem --- Dualzahlen addieren
6210 For I=1 to 3
6220 B(I)=0
6230 For J=1 to 8
6240 B(I)=B(I)+D(J,I)
6250 Next J
6260 Next I

6300 Rem --- Teilziel schon da?
6310 For I=1 to 3
6320 If B(I)/2<>Int(B(I)/2) then 6400
6330 Next I
6340 S=Int(Rnd(0)*8)+1: If A(S)=0 then 6340
6350 F=1: Goto 6600

6400 Rem --- Spalte bestimmen
6410 For I=3 to 1 step -1
6420 If B(I)/2<>Int(B(I)/2) then J=I: I=1
6430 Next I
6440 M=0
6450 For I=1 to 8
6460 If D(I,J)>0 and A(I)>M then S=I: M=A(I)
6470 Next I

6500 Rem --- Anzahl Felder bestimmen
6510 F=P(J)
6520 If J=1 then 6600
6530 If B(J-1)/2=Int(B(J-1)/2) then J=J-1: Goto 6520
6540 If D(S,J-1)>0 then F=F+P(J-1): J=J-1: Goto 6520
6550 F=F-P(J-1): J=J-1: Goto 6520

6600 Rem --- Zug ausfuehren
6610 Print "Mein Zug: Spalte ";S;" , ";F;" Felder"
6620 C(S)=C(S)-F: A(S)=A(S)-F
6630 Return
```

KAPITEL 4: TREIB DIE DAME IN DIE ECKE



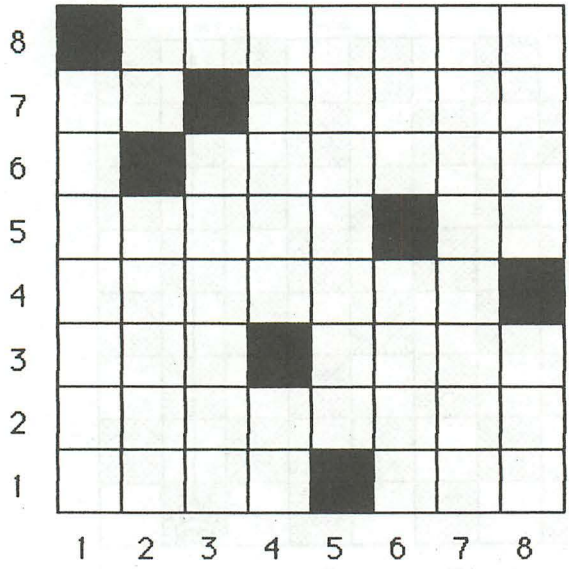
SPIELREGELN

Auf einem Schachbrett ziehen zwei Spieler abwechselnd mit einer Dame. Die Dame kann sich wie eine gewöhnliche Dame bewegen, jedoch darf sie nicht nach rechts oder nach unten ziehen. Das Spiel endet, wenn sich die Dame auf dem Feld links oben befindet. Derjenige Spieler, welcher die Dame auf dieses Feld bewegt hat, hat dann gewonnen. Zu Beginn des Spieles wird die Dame auf ein beliebiges Feld am rechten oder unteren Rand des Schachbrettes gestellt.



STRATEGIE

Die Strategie, die diesem einfachen Spiel zugrunde liegt, wurde in den letzten Kapiteln bereits gründlich erläutert. Die Strategie besteht wieder in dem Erreichen von Teilzielen. Die Teilziele können in diesem Spiel besonders leicht erkannt werden. Teilziele sind diesmal ganz bestimmte Felder, die es gilt zu erreichen. Mit ein wenig Nachdenken erhält man folgende sieben Teilziele, wobei die schwarzen Felder die Teilziele sind:



Die Felder (1,8), (2,6), (3,7), (4,3), (5,1), (6,5) und (8,4) sind also die Teilziele. Man erkennt, daß die Teilziele symmetrisch zur Diagonalen angeordnet sind.

Das folgende Programm ist einfach gestaltet und dürfte sich

selbst kommentieren. Es ist so ausgelegt, daß der Spieler immer anfängt und zu Beginn des Spieles das Anfangsfeld wählen darf.

PROGRAMMLAUF

Treib die Dame in die Ecke

Auf einem Schachbrett wird mit einer
Dame abwechselnd gezogen. Zu Beginn
wird die Dame auf ein Feld am rechten
oder unteren Rand des Brettes gestellt
Die Dame kann sich beliebig viele
Felder waagrecht nach links,
senkrecht nach oben oder diagonal
nach links oben bewegen. Sieger ist,
wer die Dame auf das Feld links oben
in der Ecke zieht.

```
Auf welchem Feld soll die Dame beginnen? 7,1
Wieviele Felder nach links? 1
Wieviele Felder nach oben? 0
Ich ziehe 2 Felder nach links und 2 Felder nach oben!
Wieviele Felder nach links? 2
Wieviele Felder nach oben? 0
Ich ziehe 0 Felder nach links und 3 Felder nach oben!
Wieviele Felder nach links? 1
Wieviele Felder nach oben? 1
Ich ziehe 0 Felder nach links und 1 Felder nach oben!

Ich habe gewonnen

Noch ein Spiel (J/N)? N
```

VARIABLENLISTE

A(X)	:x-Koordinaten der Teilziele
B(X)	:Y-Koordinaten der Teilziele
A$	:Eingabestring der Länge 1
A	:X-Koordinate der Dame
B	:Y-Koordinate der Dame
DX	:Player X-Koordinate Dame
DY	:Player Y-Koordinate Dame
PMSTART	:Startadresse PM-Daten
H	:Indikator für Gewinner
LI	:Anzahl Felder nach links bei Zug
OB	:Anzahl Felder nach oben bei Zug

Listing: Treib die Dame in die Ecke

```
1000 Rem +++ Anleitung ++++++++++++++++++++++++++++++++++++++++
1100 Rem --- Spielregeln ausgeben
1110 Graphics 0: Setcolor 2,1,10: Setcolor 1,0,2: Setcolor 4,6,4
1120 Print "Treib die Dame in die Ecke"
1130 Print
1140 Print "Auf einem Schachbrett wird mit einer"
1150 Print "Dame abwechselnd gezogen. Zu Beginn"
1160 Print "wird die Dame auf ein Feld am rechten"
1170 Print "oder unteren Rand des Brettes ge-"
1180 Print "stellt. Die Dame kann sich beliebig"
1190 Print "viele Felder waagrecht nach links,"
1200 Print "senkrecht nach oben oder diagonal"
1210 Print "nach links oben bewegen. Sieger ist,"
1220 Print "wer die Dame auf das Feld links oben"
1230 Print "in der Ecke zieht."
1240 Print
```

```
1300 Rem --- Eingabe des Anfangsfeldes
1310 Print "Auf welchem Feld soll die Dame be-"
1320 Print "ginnen ";
1330 Input A,B
1340 If A<1 or A>8 or B<1 or B>8 then 1310
1350 If A<>8 and B<>1 then 1310

1400 Rem --- Einlesen der Teilziele
1410 Dim A(7),B(7)
1420 For I=1 to 7
1430 Read C,D: A(I)=C: B(I)=D
1440 Next I

2000 Rem +++ Hauptprogramm ++++++++++++++++++++++++++++++++++++

2100 Rem --- Player Missile Graphik setzen
2110 Graphics 7: Setcolor 0,6,6: Setcolor 1,3,8: Setcolor 2,0,0:
     Setcolor 4,12,2
2120 DX=62+A*10+(B-1)*4: DY=82-B*9
2130 L=Peek(106)-24: Poke 54279,L: PMSTART=L*256
2140 Poke 559,46: Poke 623,1
2150 Poke 53277,3
2160 Poke 53248,DX
2170 For I=PMSTART+512 to PMSTART+640: Poke I,0: Next I
2180 Poke 704,14
2190 For I=PMSTART+512+DY to PMSTART+524+DY: Read J: Poke I,J:
     Next I

2200 Rem --- Schachbrett zeichnen
2210 Color 3: For I=72 to 76: Plot 20,I: Drawto 100,I: Next I
2220 For I=0 to 4: Plot 100,72+I: Drawto 132,I: Next I
2230 For I=1 to 8
2240 Plot 10+10*I,72: Drawto 42+10*I,0: Plot 20+4*I,72-9*I:Dra
     wto 100+4*I,72-9*I
2250 Next I
2260 X=12: Y=80: COL=1
2270 For I=1 to 8: Y=Y-9
2280 For J=1 to 8: X=X+10
2290 COL=3-COL: Color COL
2300 For K=X to X+8: Plot K,Y: Drawto K+2,Y-7: Next K
2310 Next J: X=X-76: COL=3-COL
2320 Next I
```

```
2400 Rem --- Spielablauf
2410 Gosub 4000                    :Rem Zug Spieler
2420 If A=1 and B=8 then H=1: Goto 2500
2430 Gosub 5000                    :Rem Zug Computer
2440 If A=1 and B=8 then H=2: Goto 2500
2450 Goto 2410

2500 Rem --- Spielende
2510 For I=15 to 0 Step -0.2: Sound 0,10+10*Rnd(0),0,I: Sound
     1,100+10*Rnd(0),16,I: Setcolor 4,12,14*Rnd(0): Next I
2520 Setcolor 4,12,2
2530 If H=1 then print "Sie haben gewonnen"
2540 If H=2 then print "Ich habe gewonnen"
2550 Print
2560 Print "Noch ein Spiel (J/N) ";
2570 Dim A$(1): Input A$
2580 If A$="J" then poke 53277,0: Run
2590 If A$="N" then End
2600 Print "Na was nun?": Goto 2560
3000 Rem +++ Dame bewegen +++++++++++++++++++++++++++++++++++
3010 If LI=0 then 3060
3020 For I=0 to LI-1
3030 Sound 0,100+(A-I)*5-B*5,30,8
3040 For J=1 to 10: Poke 53248,DX-J-10*I: Next J
3050 Next I: Sound 0,0,0,0: DX=DX-LI*10
3060 If OB=0 then Return
3070 For I=0 to OB-1
3080 Sound 0,100+A*5-(B+I)*5,30,8
3090 For K=0 to 21: Poke PMSTART+503+DY+K-9*I,Peek(PMSTART+512
     +DY+K-9*I): Next K
3100 Poke 53248,DX+4*(I+1)
3110 Next I: Sound 0,0,0,0: DX=DX+4*OB: DY=DY-9*OB
3120 Return

4000 Rem +++ Zug des Spielers ++++++++++++++++++++++++++++++++
4100 Rem --- Zugeingabe
4110 Print "Wieviele Felder nach links ";
4120 Input LI
4130 Print "Wieviele Felder nach oben ";
4140 Input OB
```

```
5200 Rem --- Verlegenheitszug
5210 LI=1: OB=0

5300 Rem --- Zug ausgeben
5310 Print "Ich ziehe ";LI;" Felder nach links und        ";
     OB;" Felder nach oben!"
5320 Gosub 3000                    :Rem Dame bewegen
5330 A=A-LI: B=B+OB
5340 Return

6000 Rem +++ Datas ++++++++++++++++++++++++++++++++++++++++
6010 Data 1,8,2,6,3,7,4,3,5,1,6,5,8,4
6020 Data 126,60,24,24,24,60,60,60,60,126,126,126,126
4200 Rem --- Zug korrekt?
4210 If LI<0 or A-LI<1 or OB<0 or B+OB>8 or LI+OB=0 then 4110
4220 If LI<>OB and LI>0 and OB>0 then 4110

4300 Rem --- Zug ausfuehren
4310 Gosub 3000                    :Rem Dame bewegen
4320 A=A-LI: B=B+OB
4330 Return

5000 Rem +++ Zug des Computers ++++++++++++++++++++++++++++

5100 Rem --- Suche nach Teilziel
5110 For I=1 to 7
5120 If A(I)=A and B(I)-B>0 then LI=0: OB=B(I)-B: Goto 5300
5130 If B(I)=B and A-A(I)>0 then LI=A-A(I): OB=0: Goto 5300
5140 If A-A(I)=B(I)-B and A-A(I)>0 then LI=A-A(I): OB=LI: Goto
     5300
5150 Next I
```

ZWEITER TEIL

KOMPLEXE SPIELE - HEURISTISCHE SUCHVERFAHREN

KAPITEL 5: BEWERTUNGSFUNKTION, SUCHVERFAHREN

EINFÜHRUNG:

Bisher hatten wir Spiele behandelt, die sich genau analysieren ließen. Durch die Analyse konnten wir jede Stellung genau charakterisieren, d.h. wir konnten unterscheiden, ob eine Spielstellung eine Gewinnstellung oder eine Verluststellung ist. Die Strategie war dadurch klar und konnte leicht auf den Computer übertragen werden.

Das Ziel besteht darin, zu jedem Spiel ein Verfahren zu entwickeln, welches entscheidet, ob eine Stellung eine Gewinn- bzw. eine Verluststellung ist. Was aber bei Spielen wie Dame, Mühle, Schach? Hier existiert kein solches Verfahren. Bei solchen Spielen müssen wir uns mit einem Notbehelf zufrieden geben. Dieser Notbehelf besteht darin, daß wir nicht mehr zwischen Gewinnstellung und Verluststellung unterscheiden, sondern den verschiedenen Stellungen Bewertungen zuweisen. Je höher die Bewertung, desto besser die Stellung. Das Verfahren, welches diese Aufgabe übernimmt, weist jeder Stellung genau einen Wert zu. Wir sprechen deshalb von einer Funktion, genau gesprochen von einer ***BEWERTUNGSFUNKTION***.

Das Prinzip der Zugauswahl besteht nun immer darin, von der vorliegenden Stellung ausgehend, durch Ausführen der Züge des Computers bzw. des Spielers die Nachfolgestellungen zu berechnen. Diese Stellungen werden dann bewertet und mit

Hilfe dieser Bewertungen wird schließlich ein Zug ausgewählt. Hierzu haben wir verschiedene Techniken zur Auswahl. Jede Technik basiert jedoch auf dem Bewerten der Stellungen. Die Bewertungsfunktion spielt deshalb eine enorm wichtige Rolle. es folgt aus diesem Grunde nun eine eingehende Untersuchung über die Bewertungsfunktion.

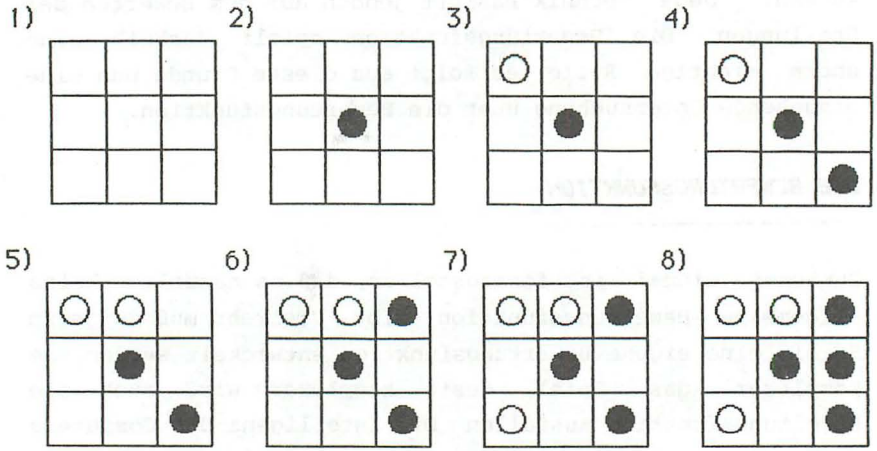
DIE BEWERTUNGSFUNKTION

Zunächst einmal ist festzustellen, daß es natürlich keine allgemeine Bewertungsfunktion gibt. Vielmehr muß zu jedem Spiel eine eigene Bewertungsfunktion entwickelt werden. Je komplexer das Spiel, desto komplexer wird auch die Bewertungsfunktion ausfallen. Die Intelligenz des Computers hängt im wesentlichen von der Güte der Bewertungsfunktionen ab. Eine optimale Bewertungsfunktion würde im Vergleich von zwei Stellungen der schlechteren Stellung immer die niedrigere Bewertung zuweisen. Leider ist diese optimale Bewertungsfunktion in der Regel nicht leicht oder gar nicht zu finden. Es geht vielmehr darum, dieser Funktion möglichst nahezukommen. Bei komplexen Spielen wie z.B. Schach ist dies äußerst schwierig, da eine Unmenge von verschiedenen Faktoren beachtet werden müssen.

Zum besseren Verständnis entwickeln wir nun eine Bewertungsfunktion zu folgendem Spiel:

Das Spiel heißt "Tic-Tac-Toe", ein anderer weit verbreiteter Name dafür ist "Kringel und Kreuze". Gespielt wird auf einem 3x3-Brett. Zwei Spieler Schwarz und Weiß setzen abwechselnd je einen schwarzen bzw. weißen Spielstein auf ein beliebiges leeres Feld. Derjenige Spieler gewinnt, welcher zuerst drei eigene Steine in einer Reihe anordnen kann. Gelingt dies keinem Spieler, so endet das Spiel mit Unentschieden. Tic-Tac-Toe ist äußerst einfach und endet bei erfahrenen Spielern immer mit Unentschieden. Für unsere Zwecke ist es jedoch bestens

geeignet. Ein möglicher Spielverlauf wäre z.B.:



Sieg von Schwarz

Wir stellen fest, daß das mittlere Feld von herausragender Bedeutung ist, da es aufgrund seiner zentralen Lage in vier verschiedenen Dreiern enthalten ist. Die Eckfelder sind am zweitwichtigsten (drei Dreier), die Seitenfelder schließlich kommen nur in zwei Dreiern vor. Die Bewertungsfunktion berücksichtigt diesen Umstand und weist den einzelnen Feldern folgende Punkte zu:

2	0	2
0	4	0
2	0	2

Die Bewertung berechnet sich nun zu Gesamtpunkte belegte Felder des Computers minus Gesamtpunkte belegte Felder des Spielers.

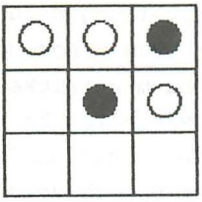
Nun werden alle auf dem Spielfeld erstellbaren Dreier durchgegangen. Dabei werden folgende Punkte verteilt:

Dreier ist noch möglich und kein Feld belegt	1 Punkte
Dreier ist möglich und ein Feld schon belegt	2 Punkte
Dreier ist möglich und zwei Felder schon belegt	4 Punkte
Dreier ist vollständig belegt	50 Punkte

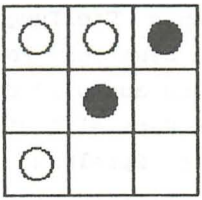
Das gilt natürlich immer sowohl für den Computer als auch für den Spieler, wobei die Punkte des Computers zur Bewertung addiert und die Punkte des Spielers von der Bewertung subtrahiert werden.

Betrachten wir als Beispiel folgende zwei Stellungen:
(Computer=Weiß, Spieler=Schwarz)

a)



b)



Wir erhalten als Bewertungen:

	Stellung a)	Stellung b)
Gesamtpunkte Felder Computer	+2	+4
Gesamtpunkte Felder Spieler	-6	-6
Punkte Dreier-Prüfung Computer	+2+1	+4+2
Punkte Dreier-Prüfung Spieler	-4-1	-2-2
	======	=======
	-6	0

Stellung a) erhält also -6 Punkte und Stellung b) erhält 0 Punkte. Stellung b) erhält somit die höhere Bewertung. Stellung b) ist auch ganz offensichtlich die günstigere Stellung für den Computer.

Soweit also zur Bewertungsfunktion. Ich will abschließend noch einmal betonen, daß es keine allgemein gültige Bewertungsfunktion gibt. Die gerade entwickelte Bewertungsfunktion ist ebenfalls nur eine einzige aus vielen möglichen. Es könnte genausogut eine andere Funktion zu Tic-Tac-Toe entwickelt werden. Wichtig bei der Wahl einer Bewertungsfunktion ist die Qualität der Bewertungen und die Zeit, die zum Bewerten einer Stellung benötigt wird. Es gilt, bei einer möglichst geringen Zeit eine möglichst hohe Qualität der Bewertungen zu erreichen.

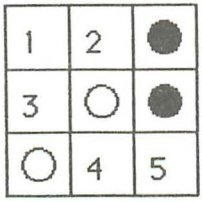
EINFACHES SUCHVERFAHREN MIT SUCHTIEFE 1

Die Aufgabe eines Suchverfahrens besteht darin, von einer gegebenen Stellung aus Nachfolgestellungen zu generieren und zu bewerten. Formal ausgedrückt: Von einer vorgegebenen Stellung aus wird der weitere Spielbaum entwickelt, welchen wir in Kapitel 1 eingeführt haben. Natürlich kann nicht der gesamte Spielbaum generiert werden, das würde, zumal in Basic, viel zu viel Zeit kosten. Der Spielbaum wird deshalb nur bis zu einer bestimmten vorgegebenen Tiefe generiert.

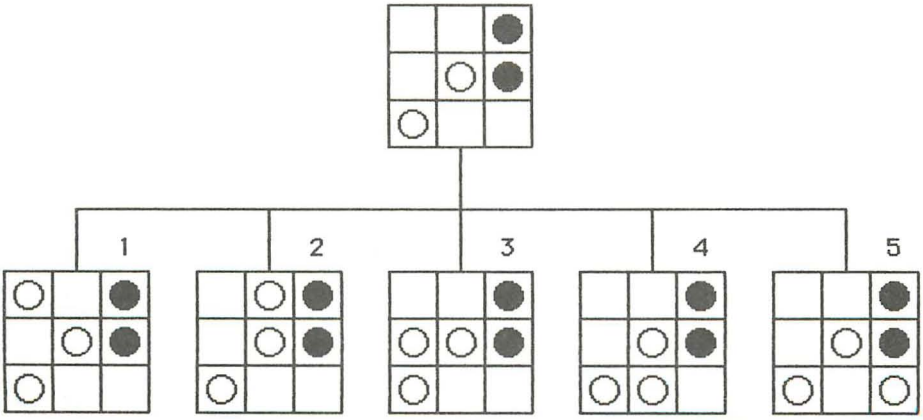
Am einfachsten ist es, wenn nur die unmittelbaren Nachfolgestellungen entwickelt werden. Wir sprechen dann von einem Suchverfahren mit der Suchtiefe 1. Der Vorteil eines solchen Verfahrens liegt darin, daß nur sehr wenige Stellungen generiert werden müssen. Dadurch wird die Denkzeit des Computers entsprechend niedrig gehalten. Dieser Faktor ist sehr wichtig, da in Basic aufgrund der niedrigen Arbeitsgeschwindigkeit die Denkzeit des Computers schnell das erträgliche Maß überschritten hat. Der Nachteil

besteht in der geringen Voraussicht des Computers, da ja nur die unmittelbaren Nachfolgestellungen gebildet werden. Bei einem solchen Suchverfahren mit der Suchtiefe 1 ist eine gute Bewertungsfunktion deshalb unerläßlich.

Als Beispiel betrachten wir das auf den vorhergehenden Seiten vorgestellte Spiel "Tic-Tac-Toe" und die dazu entwickelte Bewertungsfunktion. Nehmen wir also an, der Computer (Weiß) sei bei folgender Stellung am Zuge:



Der Computer hat nun die Möglichkeit, seinen Stein auf eines der fünf leeren durchnumerierten Felder zu setzen. Dem Computer stehen also fünf verschiedene Züge zur Auswahl. Der Computer führt nun diese Züge durch und bewertet die so entstandenen Stellungen. In unserem Beispiel ergäbe das folgende Bewertungen:



	Zug 1	Zug 2	Zug 3	Zug 4	Zug 5
Bewertung:	+14	+10	+10	+10	+16

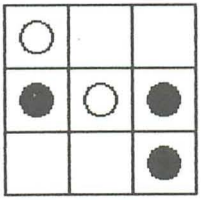
Der Computer erhält also diese fünf aufgeführten Bewertungen. Der Zug mit der höchsten Bewertung wird nun ausgewählt, in diesem Falle also Zug Nummer 5.

Sie sehen, dieses Verfahren mit der Suchtiefe 1 erweist sich als äußerst wirkungsvoll. Wir werden dieses Verfahren nun verallgemeinern und auch auf größere Suchtiefen ausdehnen.

DIE MINIMAX-PROZEDUR

Auf den vorhergehenden Seiten haben wir ein einfaches Suchverfahren mit der Suchtiefe 1 kennengelernt. Bei diesem Suchverfahren wurden sämtliche Züge des Computers ausgeführt und die entstandenen Stellungen mit Hilfe einer Bewertungsfunktion bewertet. Die möglichen folgenden Züge des Spielers wurden dabei nicht berücksichtigt.

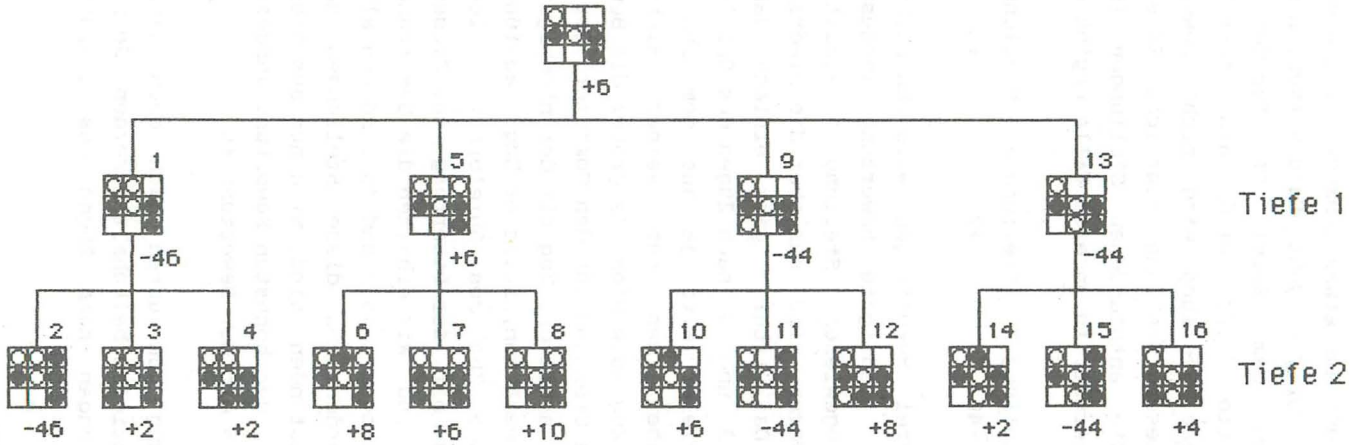
Wenn nach jedem Zug des Computers noch sämtliche Züge des Spielers ausgeführt werden, so sprechen wir von der Suchtiefe 2. Betrachten wir dazu folgende Stellung im Spiel "Tic-Tac-Toe":



Der Computer (Weiß) sei nun am Zuge. Er hat vier Züge zur Auswahl. Nach jedem Zug des Computers stehen wiederum drei Züge des Spielers zur Auswahl. Bei der Suchtiefe 2 müßte der Computer also 16 Stellungen generieren. Betrachten Sie dazu Bild 1.

Die 16 Stellungen wurden durchnumeriert, wobei diese Numerierung die Reihenfolge der Abarbeitung angibt. Der Computer führt seine Züge also von links nach rechts und

von oben nach unten durch. Zuerst wurde also Stellung 1, dann Stellung 2 usw. generiert. Die 12 Stellungen der Tiefe 2 sind Endstellungen. Nur sie werden bewertet. Ausgehend von diesen Bewertungen werden nun den darüberliegenden Stellungen, d.h. den Elternstellungen, Bewertungen zugewiesen. Aus Stellung 1,5,9 und 13 wird anschließend die Stellung mit der höchsten Bewertung ausgesucht.


+6
1
5
9
13
-46
+6
-44
-44
Tiefe 1
2
3
4
6
7
8
10
11
12
14
15
16
-46
+2
+2
+8
+6
+10
+6
-44
+8
+2
-44
+4
Tiefe 2

Schauen wir uns nun etwas genauer an, wie dieses Bewerten im Einzelnen vor sich geht. Dazu verwenden wir wieder die von uns zu dem Spiel "Tic-Tac-Toe" entwickelte Bewertungsfunktion. Es wird also zuerst Stellung 1 gebildet. Diese Stellung wird nicht bewertet. Vielmehr werden zu dieser Stellung sämtliche Züge des Spielers ausgeführt. Die entstandenen Stellungen (Endstellungen) werden nun bewertet. In unserem Falle ergibt das:

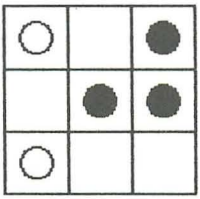
	Stellung 2	Stellung 3	Stellung 4
Bewertung:	-46	+2	+2

Aus diesen drei Bewertungen wird nun nicht die höchste, sondern die niedrigste Bewertung ausgesucht und der Stellung 1 zugewiesen. Stellung 1 erhält als Bewertung somit -46. Warum wird nun aber die niedrigste Bewertung ausgewählt? Das rührt ganz einfach daher, daß die Stellungen 2,3 und 4 nach Zügen des Spielers entstanden sind. Der Spieler wird ja aus dem ihm zur Verfügung stehenden Zügen den aus seiner Sicht am besten erscheinenden Zug auswählen. Je größer die Bewertung, desto besser ist die Stellung für den Computer. Der beste Zug des Spielers ist somit der Zug mit der niedrigsten Bewertung. Der Computer weiß nun, falls er Zug 1 ausführt, ergibt der darauf folgende Zug des Spielers im schlechtesten Falle eine Stellung mit der Bewertung -46. Dieses Prinzip wird weitergeführt und wir erhalten die Bewertungen gemäß Bild 1. Den Stellungen 1,5,9 und 13 sind nun also Bewertungen zugewiesen worden. Da diese Stellungen nach Zügen des Computers entstanden sind, wird nun aus diesen Stellungen diejenige mit der höchsten Bewertung ausgesucht. Das wäre also Stellung 5 mit der Bewertung +6.

Diese Bewertung von unten nach oben kann nun auch bei größeren Suchtiefen beibehalten werden. Je nachdem, ob die Nachfolgestellungen nach Zügen des Computers bzw. des

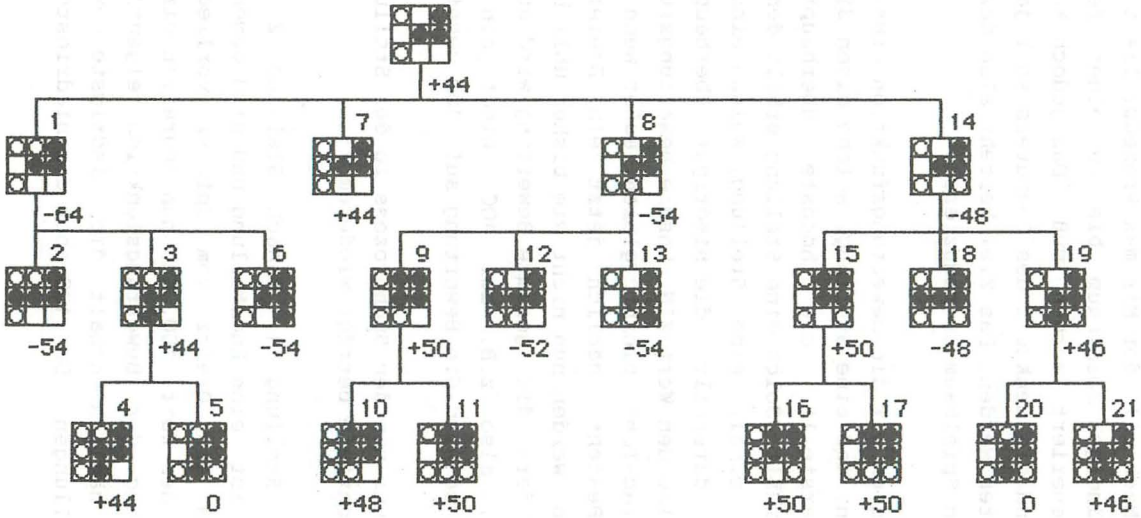
Spielers entstanden sind, wird die höchste bzw. die niedrigste Bewertung ausgesucht und der Elternstellung zugewiesen. Es wird also einmal das Minimum und einmal das Maximum ausgewählt. Daher der Name "Minimax-Prozedur". Numerieren wir die Suchtiefen von oben nach unten durch, so wird bei ungeraden Suchtiefen das Maximum und bei geraden Suchtiefen das Minimum ausgewählt.

Soweit so gut. Der Spielbaum wird also bis zu der geforderten Suchtiefe entwickelt und den Stellungen werden von unten nach oben Bewertungen zugewiesen. Wir müssen nun noch den Fall eines Abbruches besprechen. Ein Abbruch liegt dann vor, wenn zu einer Stellung keine Nachfolgestellungen entwickelt werden, obwohl die geforderte Suchtiefe noch nicht erreicht ist. Überlegen Sie selber einmal, wie eine solche Stellung beschaffen sein muß! Wir kommen zu dem Ergebnis, daß ein Abbruch vorliegt, wenn zu einer Stellung keine Nachfolgestellungen existieren, da zu dieser Stellung keine Züge mehr möglich sind oder aber wenn eine Stellung einen Sieg für den Computer bzw. den Spieler darstellt. Kurz gesagt also erfolgt bei einer Stellung dann ein Abbruch, wenn bei dieser Stellung das Spiel zu Ende ist. Das leuchtet ja auch unmittelbar ein. Das Vorgehen bei einem solchen Abbruch ist ganz einfach: Die Stellung, bei der der Abbruch erfolgt, wird mit Hilfe der Bewertungsfunktion bewertet und das Verfahren geht in gewohnter Weise weiter. Untersuchen Sie das Vorgehen des Computers zu folgender Stellung, wobei der Computer (Weiß) am Zuge ist und die Minimax-Prozedur mit der Suchtiefe 3 anwendet. Das Ergebnis finden Sie in Bild 2 dargestellt.



Der Computer muß 21 Stellungen generieren. Würde in keinem Falle ein Abbruch erfolgen, so müßten in Tiefe eins 4, in Tiefe zwei 12 und in Tiefe drei 24 Stellungen gebildet werden. Das wären insgesamt also 40 Stellungen. Die Anzahl der zu entwickelnden Stellungen wird durch das Auftreten von vorzeitigen Abbrüchen also beträchtlich verringert.

Diese Methode des Abbruches läßt sich nun noch verfeinern. Wir gelangen dann zu einem neuen Suchverfahren, welches Alpha-Beta Pruning genannt wird.


+44
1
7
8
14
-64
+44
-54
-48
2
3
6
9
12
13
15
18
19
-54
+44
-54
+50
-52
-54
+50
-48
+46
4
5
10
11
16
17
20
21
+44
0
+48
+50
+50
+50
0
+46
Tiefe 1
Tiefe 2
Tiefe 3

ALPHA-BETA PRUNING

Der große Nachteil der Minimax-Prozedur liegt darin, daß der vollständige Spielbaum bis zu einer festgelegten Suchtiefe generiert werden muß. Das jedoch kostet enorm viel Zeit und die Denkzeit des Computers soll ja möglichst gering gehalten werden. Das Ziel besteht also darin, den zu generierenden Spielbaum zu reduzieren.

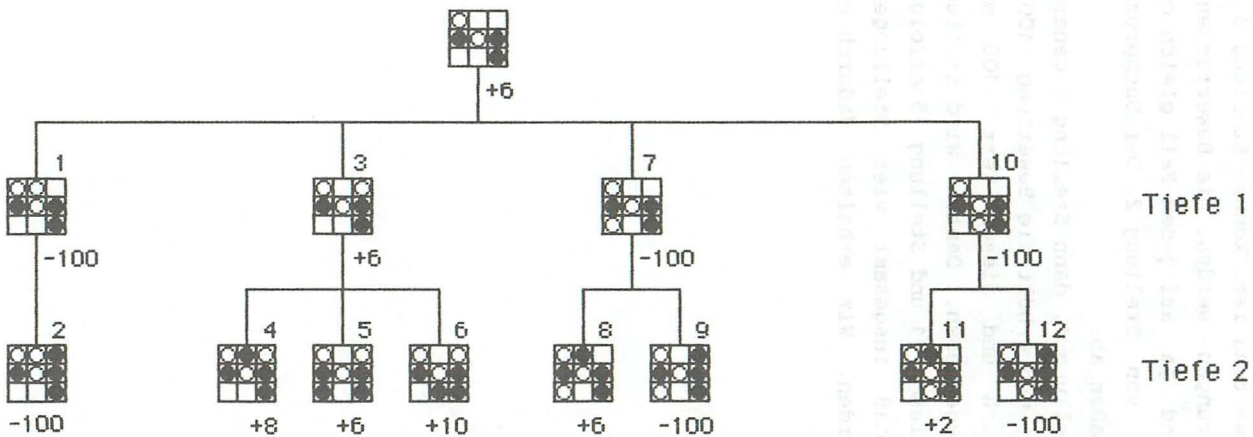
Zunächst müssen wir die Bewertungsfunktion etwas umändern. Wir verlangen, daß eine Stellung, welche einen Sieg für den Computer darstellt, die höchste überhaupt mögliche Bewertung erhält. Solch eine Stellung erhält den Wert MAX. Entsprechend erhält eine Stellung, welche einen Sieg für den Spieler darstellt, die niedrigste überhaupt mögliche Bewertung, also den Wert MIN. Unsere Bewertungsfunktion zum Spiel "Tic-Tac-Toe" bleibt gleich, außer wenn ein Dreier vorliegt. Besteht nämlich jetzt ein Dreier für den Computer, so werden nun nicht wie bisher üblich 50 Punkte addiert, sondern die gesamte Bewertung wird auf den Wert MAX gesetzt, also z.B. auf 100. Liegt ein Dreier vom Spieler vor, so wird die Bewertung auf -100 gesetzt.

Schauen wir uns nun den Suchprozess zu der Stellung in Bild 1 an, die Suchtiefe beträgt wiederum 2:

Zuerst wird Stellung 1, danach Stellung 2 generiert. Stellung 2 ist eine Endstellung und wird dementsprechend bewertet. Da ein Dreier vom Spieler vorliegt, erhält Stellung 2 den Wert -100. Und nun begreifen wir auch, was diese Änderung der Bewertungsfunktion eigentlich soll. Stellung 1 nämlich erhält die niedrigste Bewertung der Nachfolgestellungen. Da -100 aber die niedrigste überhaupt

vorkommende Bewertung ist, können Stellung 3 und Stellung 4 völlig übersprungen werden. Die Bewertungen dieser beiden Stellungen sind ja auf jeden Fall gleich oder größer als die Bewertung von Stellung 2. Der Suchprozess läuft also nun folgendermaßen ab:

Zuerst wird Stellung 1, dann Stellung 2 generiert. Stellung 2 wird bewertet und erhält die Bewertung -100. Hier erfolgt nun ein Abbruch und dieser Wert -100 wird sofort der Stellung 1 zugewiesen. Danach wird Stellung 5 generiert usw. In Stellung 11 und Stellung 15 erfolgt wiederum ein Abbruch, so daß insgesamt vier Stellungen (3,4,12,16) eingespart werden. Wir erhalten dadurch einen Spielbaum gemäß Bild 3.


+6
1
3
7
10
Tiefe 1
-100
+6
-100
-100
2
4
5
6
8
9
11
12
Tiefe 2
-100
+8
+6
+10
+6
-100
+2
-100

Alpha-Beta Pruning geht jetzt sogar noch weiter. Betrachten wir dazu Bild 3. Stellung 1 erhält den Wert -100, Stellung 3 den Wert +6 und Stellung 7 wiederum den Wert -100. Bis zu dieser Stelle ändert sich nichts. Nun wird Stellung 10 und Stellung 11 generiert. Stellung 11 erhält den Wert +2. Nun ist Stellung 12 an der Reihe. Doch halt, muß Stellung 12 wirklich generiert und bewertet werden? Die Antwort lautet NEIN! Wir wissen, daß Stellung 10 den niedrigsten Wert der Nachfolgestellungen erhält. Stellung 10 erhält somit einen Wert, der kleiner oder gleich dem Wert von Stellung 11 ist, also einen Wert nicht größer als +2. Stellung 3 jedoch hat den Wert +6. Da aus Stellung 1,3,7 und 10 die höchste Bewertung ausgesucht wird, kann Stellung 12 keine Änderung mit sich bringen. In Stellung 11 erfolgt deshalb ein Abbruch. Stellung 10 erhält sofort den Wert +2 zugewiesen und Stellung 12 bleibt unberücksichtigt. Es wurde dadurch also wiederum eine Stellung eingespart.

Der Abbruch erfolgte im vorigen Beispiel bei einer gerade Suchtiefe. Das kann jedoch auch bei ungeraden Suchtiefen der Fall sein. Bild 2 macht dies deutlich. Stellung 4 erhält den Wert +44. Dieser Wert ist deutlich größer als der Wert von Stellung 2. Stellung 3 erhält den höchsten Wert der Nachfolgestellungen, also einen Wert größer oder gleich +44. Aus Stellung 2 und 3 wird wiederum der niedrigste Wert ausgesucht. Da der Wert von Stellung 2 auf jeden Fall kleiner als der Wert von Stellung 3 ist, kann Stellung 5 unberücksichtigt bleiben.

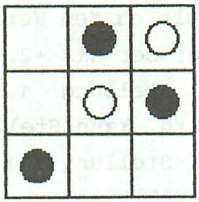
Wir haben somit zwei verschiedene Abbrüche kennengelernt. Zusammenfassend läßt sich sagen:

-Ist der Wert einer Stellung aus einer geraden Suchtiefe kleiner als der Wert einer Stellung aus der vorangehenden Suchtiefe, so wird die Suche an dieser Stelle abgebrochen.

Wir sprechen in diesem Falle von einem Alpha-Abbruch.

-Ist der Wert einer Stellung aus einer ungeraden Suchtiefe größer als der Wert einer Stellung aus der vorangehenden Suchtiefe, so erfolgt ein Beta-Abbruch.

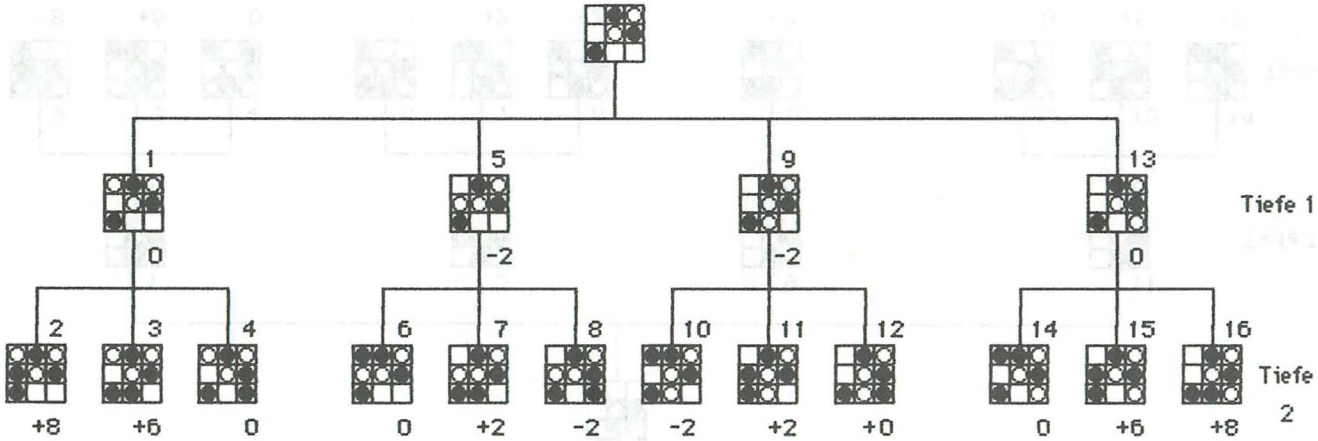
Wenden wir die gesammelten Ergebnisse nun zur Vertiefung auf folgende Stellung an:

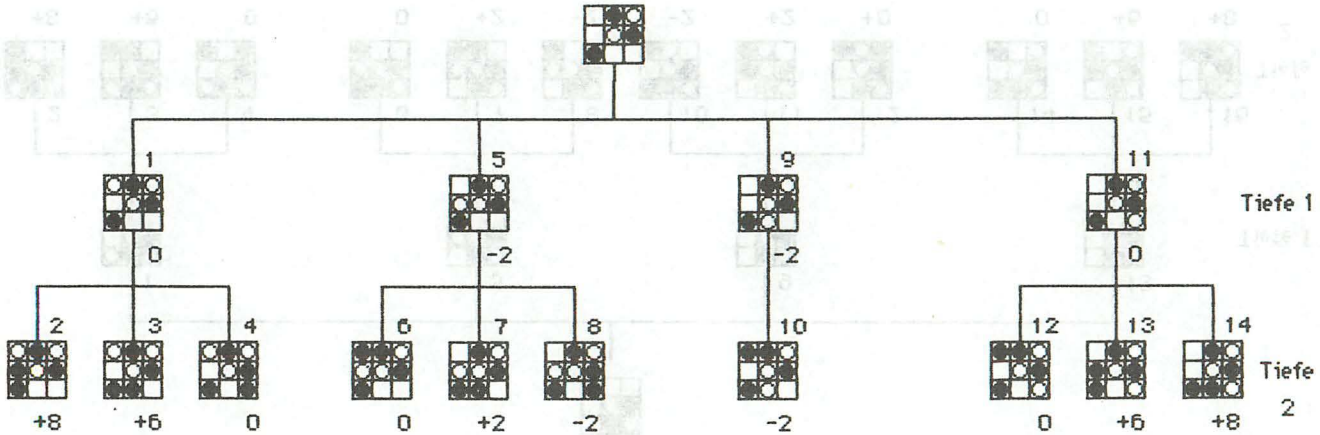


Der Computer sei Weiß, der Spieler sei Schwarz. Der Computer ist am Zuge und führt die Züge wie gewohnt von links nach rechts und von oben nach unten durch. Zeichnen Sie zur Überprüfung Ihrer Kenntnisse nun den entstehenden Spielbaum zur Minimax-Prozedur und zu Alpha-Beta Pruning, jeweils mit Suchtiefe 2!

Das Ergebnis finden Sie auf Bild 4 und Bild 5 dargestellt. Bei der Minimax-Prozedur (Bild 4) müssen 16 Stellungen generiert werden. Stellung 1 und Stellung 13 erhalten die gleichen Bewertungen. Der Computer wird aus diesen beiden Stellungen also eine auswählen. Bei Alpha-Beta Pruning gilt dies natürlich genauso. Hier müssen jedoch nur 14 Stellungen generiert werden, da in Stellung 10 ein Alpha-Abbruch erfolgt. Die Bewertung von Stellung 10 (-2) ist nämlich kleiner als die Bewertung von Stellung 1, wodurch Stellung 9 sofort die Bewertung -2 zugewiesen werden kann.

In den nächsten Kapiteln wird das in diesem Kapitel erarbeitete theoretische Wissen nun auf praktische Beispiele angewendet!

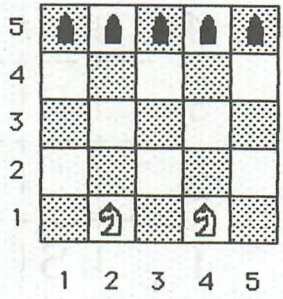

1
5
9
13
Tiefe 1
0
-2
-2
0
2
3
4
6
7
8
10
11
12
14
15
16
Tiefe 2
+8
+6
0
0
+2
-2
-2
+2
+0
0
+6
+8


1
5
9
11
Tiefe 1
0
-2
-2
0
2
3
4
6
7
8
10
12
13
14
Tiefe 2
+8
+6
0
0
+2
-2
-2
0
+6
+8

```
                 5  B |   | B | S |
                    +---+---+---+---
SPRINGER         4    |   |   |   | B
                    +---+---+---+---
GEGEN            3    | B |   |   |
                    +---+---+---+---
BAUER            2    |   |   |   |
                    +---+---+---+---
                 1    | S |   |   |
                      1   3   3   4   5

Sie sind am Zuge: von ?3,3
                  nach?4,5
Mein Zug: von 2,4  nach 2,3
Sie sind am Zuge: von ?■
```

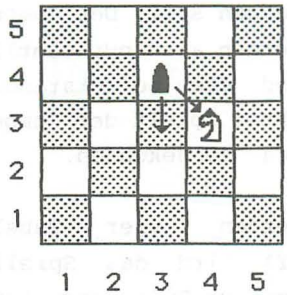
SPIELREGELN

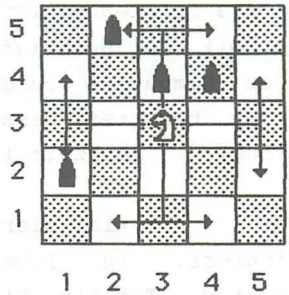
Spieler Weiß und Spieler Schwarz kämpfen gegeneinander. Zu Beginn hat Weiß zwei Springer und Schwarz fünf Bauern zur Verfügung. Die Figuren werden folgendermaßen auf einem 5 X 5 Brett aufgestellt:



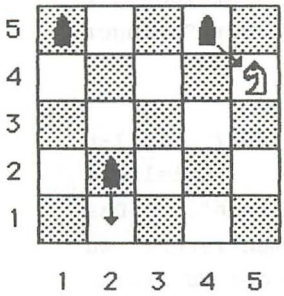
Die Bauern können sich nur vorwärts bewegen, also in Richtung von Zeile 1. Ein Bauer kann geradeaus auf ein angrenzendes leeres Feld ziehen oder aber diagonal auf ein angrenzendes Feld ziehen, falls sich auf diesem Feld ein Springer befindet. Der Springer gilt in diesem Falle als geschlagen und wird vom Brett entfernt.
Die Springer ziehen zunächst zwei (auch besetzte) Felder nach oben, unten, rechts oder links, drehen sich dann um 90 Grad und ziehen auf das angrenzende Feld. Falls dieses Feld von einem Bauern besetzt ist, wird der geschlagene Bauer vom Brett entfernt. Ein Springer darf nicht auf ein vom zweiten Springer besetztes Feld ziehen.

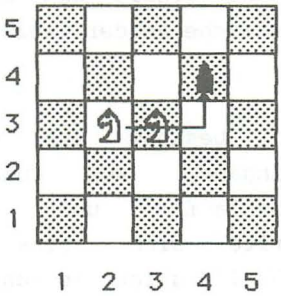
Folgende Stellungen verdeutlichen die Zugmöglichkeiten von Weiß bzw. Schwarz:

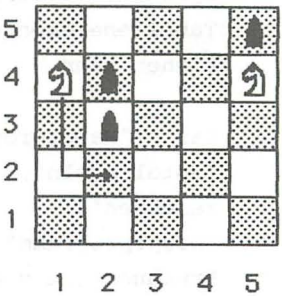




Weiß gewinnt, falls er alle Bauern von Schwarz geschlagen hat oder falls die Bauern nicht mehr ziehen können.
Schwarz gewinnt, falls er einen Bauern in Zeile 1 bringt oder falls er die zwei Springer geschlagen hat.
Folgende Züge führen z.B. jeweils zum Ende des Spieles:







PROGRAMMERLÄUTERUNG

Der Computer übernimmt die Rolle von Schwarz (Bauern), der Spieler die Rolle von Weiß (Springer). Der Spieler kann zu

Beginn des Spieles wählen, wer anfangen soll. Der Computer rechnet mit der Suchtiefe 1. Es werden also nur sämtliche Züge des Computers ausgeführt und die so entstandenen Stellungen bewertet. Dessen ungeachtet spielt der Computer vorzüglich, die Denkzeit liegt bei ca. 10 Sekunden.

In der "Initialisierung" werden vier Tabellen dimensioniert. In Tabelle A(X,Y) wird das Spielfeld festgehalten, Tabelle B(X,Y) dient zum Zwischenspeichern des Spielfeldes. Die Tabellen X(X) und Y(X) werden zum Ermitteln der möglichen Springerzüge benötigt. Die Variable H zeigt an, wer am Zuge ist. Bei H=1 ist der Spieler am Zuge, bei H=2 entsprechend der Computer. In den Zeilen 1200 bis 1270 wird das Spielfeld gesetzt. Ein leeres Feld wird durch eine Null gekennzeichnet, ein Feld mit einem Springer durch eine Eins und ein Feld mit einem Bauern durch eine Zwei. Nach dem Setzen des Spielfeldes werden die Tabellen X(X) und Y(X) mit den Werten der DATA-Liste in Zeile 8010 initialisiert. Ist (X,Y) ein Feld, auf dem sich ein Springer befindet, so erhält man durch Addition der Tabellenelemente sämtliche Felder, auf welche der Springer ziehen kann.

Das "Hauptprogramm" besteht aus den fünf Teilen "Spielregeln", "Eingabe, wer anfängt", "Spielfeld zeichnen", "Spielablauf" und "Spielende". Das "Hauptprogramm" dürfte sich selbst kommentieren, zu Erwähnen ist vieleicht lediglich die Funktion der Variablen "AUS". In Zeile 2530, also nach dem Spieler- bzw. Computerzug wird das Unterprogramm in Zeile 6000 aufgerufen. Dieses Unterprogramm testet, ob das Spiel zu Ende ist. Wenn ja wird AUS gleich Eins gesetzt, ansonsten AUS=0.

Die Programmteile "Zug XV,YV nach XN,YN ausführen" und "Zug des Spielers" bedürfen ebenfalls keines weiteren Kommentars. Zu Erklären ist nun natürlich noch der Teil "Zug des Computers":

Der "Zug des Computers" befindet sich in den Zeilen 5000 bis 5340 und besteht aus zwei Teilen. Im ersten Teil wird der Zug mit der höchsten Bewertung ausgewählt, im zweiten Teil wird dieser Zug ausgeführt und ausgegeben.

In Zeile 5110 wird das Spielfeld abgespeichert. Dann wird die Variable MAX auf -100 gesetzt. Diese Variable dient zum Speichern der höchsten Bewertung. Da die Bewertungen immer größer als -100 sind, wird also immer ein Zug ausgewählt. In Zeile 5130 werden zwei ineinander geschachtelte Schleifen VY und VX gestartet. Mit diesen zwei Schleifen wird das gesamte Spielfeld abgetastet. Wichtig sind alleine die Felder mit Bauern. Bei einem Feld ohne Bauer werden die Schleifen sofort weitergezählt (Zeile 5140). Trifft der Computer nun auf ein Feld mit einem Bauern, so wird getestet, ob dieser Bauer ziehen kann. Ist ein Zug möglich, so wird das Unterprogramm "Bewertung" aufgerufen. Der mögliche Zug wird mit den Variablen VX, VY, NX und NY übergeben. In diesem Unterprogramm wird nun zunächst der Zug ausgeführt, d.h. A(VX,VY)=0 und A(NX,NY)=2. Danach wird die Stellung bewertet und das Spielfeld wird wieder neu gesetzt. Ist die Bewertung größer MAX, so wird MAX=BEW gesetzt und dieser Zug wird in den Variablen Z1-Z4 abgespeichert. Diese Variablen geben nach Ausführung sämtlicher Züge des Computers also den Zug mit der höchsten Bewertung an. Haben zwei Stellungen die gleiche Bewertung, so spielt der Zufallsfaktor in Zeile 7220 eine Rolle.

Die Bewertung einer Stellung erfolgt nach folgenden Kriterien:

- Ein Feld mit einem Springer ergibt 10 Punkte Abzug. Je weniger Springer, desto besser.

-Jeder Bauer, den ein Springer schlagen kann, ergibt 5 Punkte Abzug. Ist dieser Bauer einmal gedeckt, so werden anschließend 2 Punkte addiert, ist der Bauer zweimal gedeckt, so werden 4 Punkte addiert. Insgesamt ergibt das

also:

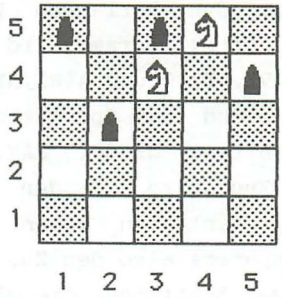
Schlagen eines ungedeckten Bauern möglich: -5 Punkte
Schlagen eines einmal gedeckten Bauern möglich: -3 Punkte
Schlagen eines zweimal gedeckten Bauern möglich: -1 Punkte

-Befindet sich ein Bauer in Zeile 1, so wird die Bewertung auf 500 gesetzt (maximale Bewertung).

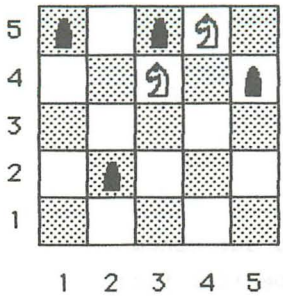
-Kann ein Bauer vorwärts auf ein angrenzendes leeres Feld ziehen, so wird die Bewertung um 1 erhöht.

-Kann ein Bauer einen Springer schlagen, so werden 2 Punkte addiert.

Zur Veranschaulichung berechnen wir die Bewertungen der Züge des Computers zu folgender Stellung:



Der Computer hat drei Züge zur Auswahl: (2,3) nach (2,2), (5,4) nach (5,3) und (1,5) nach (1,4). Zuerst wird Zug (2,3) nach (2,2) ausgeführt. Das Spielfeld hat dann folgendes Aussehen:



Diese Stellung erhält nun folgende Bewertung:

2 Springer:	-20 Punkte
2 ungedeckte Bauern können geschlagen werden:	-10 Punkte
3 Bauern können ein Feld nach vorne ziehen:	+3 Punkte

Insgesamt erhält diese Stellung somit -27 Punkte.

Auf gleiche Weise werden die Bewertungen für die beiden anderen Züge ermittelt. Wir erhalten:

(2,3) nach (2,2)	BEW=-27 Punkte
(5,4) nach (5,3)	BEW=-23 Punkte
(1,5) nach (1,4)	BEW=-17 Punkte

Der Computer würde also Zug (1,5) nach (1,4) wählen, was zweifellos der beste Zug ist.

PROGRAMMLAUF

"Springer gegen Bauer"

"Springer gegen Bauer" wird auf einem 5 X 5 Brett gespielt. Ich besitze fuenf Bauern, Sie besitzen zwei Springer. Die Bauern koennen geradeaus auf ein angrenzendes leeres Feld ziehen oder einen Springer schlagen, der sich auf einem angrenzenden Feld diagonal vorwaerts befindet. Springer ziehen wie im Schach. Ein Springer darf sich nicht auf ein vom zweiten Springer besetztes Feld bewegen. Falls ein Bauer die gegenüberliegende Seite erreicht oder falls ein Spieler nicht mehr ziehen kann, endet das Spiel.

```
Wollen Sie anfangen (J/N)? J

Sie sind am Zuge: Von? 2,1
                  Nach? 4,2

Mein Zug: Von 2,5 nach 2,4

Sie sind am Zuge: Von? 4,1
                  Nach: 2,2

Mein Zug: Von 4,5 nach 4,4

Sie sind am Zuge: Von? 2,2
                  Nach? 3,4
```

```
Mein Zug: Von 1,5 nach 1,4

Sie sind am Zuge: Von? 3,4
                  Nach? 5,5

Mein Zug: Von 1,4 nach 1,3

Sie sind am Zuge: Von? 4,2
                  Nach? 2,3

Mein Zug: Von 3,5 nach 3,4

Sie sind am Zuge: Von? 5,5
                  Nach? 3,4

Mein Zug: Von 4,4 nach 4,3

Sie sind am Zuge: Von? 2,3
                  Nach? 4,2

Mein Zug: Von 1,3 nach 1,2

Sie sind am Zuge: Von? 3,4
                  Nach? 2,2

Mein Zug: Von 1,2 nach 1,1

Ich habe gewonnen!

Noch ein Spiel (J/N)? N

Ready
```

VARIABLENLISTE

A(X,Y)	:Tabelle für Festhalten des Spielfeldes
B(X,Y)	:Zwischenspeichern des Spielfeldes
X(X),Y(X)	:Tabellen zum Ermitteln der möglichen Springerzüge
A$	:Eingabestring der Länge Eins
H	:Indikator für am Zug befindlicher Spieler
AUS	:Variable, die anzeigt, ob das Spiel zu Ende ist. AUS=0 Spiel ist nicht zu Ende AUS=1 Spiel ist zu Ende
XV,YV	:Koordinaten des Feldes, auf dem sich die Figur, mit der gezogen wird, befindet
XN,YN	:Koordinaten des Feldes, auf das die Spielfigur ziehen soll
MAX	:Festhalten der maximalen Bewertung
VX,VY	:Laufvariablen zum Abtasten des Spielfeldes
X,Y	:Laufvariablen zum Abtasten des Spielfeldes
Z1,Z2,Z3,Z4	:Bezeichnet den Computerzug
BEW	:Bewertung

Listing: Springer gegen Bauer

```
1000 Rem +++ Initialisierung +++++++++++++++++++++++++++++++++

1100 Rem --- Bildschirm, Dimensionierung
1110 Graphics 0: Setcolor 2,2,2
1120 Dim A(5,5),B(5,5),X(8),Y(8),A$(1)

1200 Rem --- Spielfeld setzen
1210 For I=1 to 5
1220 For J=1 to 5
1230 If I=5 then A(J,I)=2: Goto 1250
1240 A(J,I)=0
1250 Next J
1260 Next I
1270 A(2,1)=1: A(4,1)=1

1300 Rem --- Tabellen Springerzuege einlesen
1310 For I=1 to 8
1320 Read X,Y: X(I)=X: Y(I)=Y
1330 Next I

2000 Rem +++ Hauptprogramm +++++++++++++++++++++++++++++++++++

2100 Rem --- Spielregeln
2110 Print
2120 Print "•• Springer gegen Bauer ••"
2130 Print
2140 Print "'Springer gegen Bauer' wird auf einem"
2150 Print "5x5 Brett gespielt. Ich besitze fuenf"
2160 Print "Bauern, sie besitzen zwei Springer."
2170 Print "Die Bauern koennen geradeaus auf ein"
2180 Print "angrenzendes leeres Feld ziehen oder"
2190 Print "einen Springer schlagen, der sich"
2200 Print "auf einem angrenzenden Feld diagonal"
2210 Print "vorwaerts befindet. Springer ziehen"
2220 Print "wie im Schach. Ein Springer darf sich"
2230 Print "nicht auf ein vom zweiten Springer"
2240 Print "besetztes Feld bewegen. Falls ein"
2250 Print "Bauer die gegenueberliegende Seite"
2260 Print "erreicht oder falls ein Spieler nicht"
2270 Print "mehr ziehen kann, endet das Spiel."
2280 Print
```

```
2300 Rem --- Eingabe, wer anfaengt
2310 Print "Wollen Sie anfangen (J/N) ";
2320 Input A$
2330 If A$="J" then H=1: Goto 2400
2340 If A$="N" then H=2: Goto 2400
2350 Print "Bist du taub, ich sagte 'J' oder 'N'": Goto 2310

2400 Rem --- Spielfeld zeichnen
2410 Graphics 2: Setcolor 0,8,6: Setcolor 2,2,2: Setcolor 3,3,12
2420 Position 0,2: Print #6;"Springer": Position 1,4: Print #6;
     "gegen": Position 1,6: Print #6;"Bauer"
2430 For I=17 to 21: Color I: Plot 9,10-2*(I-16): Plot 8+2*(I-16),
     9: Next I
2440 For I=1 to 7 step 2: Position 10,I: Print  6;"-+-+-+-+-";:
     Next I
2450 Color 33: For I=11 to 17 step 2: For J=0 to 8 step 2: Plot
     I,J: Next J: Next I
2460 Color 226: For I=10 to 18 step 2: Plot I,0: Next I
2470 Color 243: Plot 12,8: Plot 16,8

2500 Rem --- Spielablauf
2510 If H=1 then gosub 4000          :Rem Spielerzug
2520 If H=2 then gosub 5000          :Rem Computerzug
2530 Gosub 6000                      :Rem Spielende?
2540 If AUS=0 then H=3-H: Goto 2510

2600 Rem --- Spielende
2610 For I=15 to 0 step -1: For J=30 to 0 step -5: So.0,J,10,I:N.J:N.I
2620 If H=1 then print "Sie haben gewonnen!"
2630 If H=2 then print "Ich habe gewonnen!"
2640 Print
2650 Print "Noch ein Spiel (J/N) ";
2660 Input A$
2670 If A$="J" then Run
2680 End

3000 Rem +++ Zug XV,YV nach XN,YN ausfuehren ++++++++++++++++++++
3010 U=A(XV,YV)
3020 Color 0: Plot 8+2*XV,10-2*YV
3030 Color 226: If U=1 then Color 243
3040 Plot 8+2*XN,10-2*YN
3050 Return
```

```
4000 Rem +++ Zug des Spielers ++++++++++++++++++++++++++++++++++

4100 Rem --- Eingabe des Zuges
4110 Print "Sie sind am Zuge:  Von ";
4120 Input XV, YV
4130 If XV<1 or YV<1 or XV>5 or YV>5 then 4110
4140 Print "                    Nach ";
4150 Input XN, YN
4160 If XN<1 or YN<1 or XN>5 or YN>5 then 4140

4200 Rem --- Ist Zug korrekt?
4210 If A(XV,YV)<>1 or A(XN,YN)=1 then print "Ungueltiger Zug!!":
     Goto 4100
4220 If Abs(XV-XN)+Abs(YV-YN)<>3 then print "Ein Springer kann so
     nicht ziehen!!": Goto 4100

4300 Rem --- Zug ausfuehren
4310 Gosub 3000: A(XV,YV)=0: A(XN,YN)=1
4320 Return

5000 Rem +++ Zug des Computers ++++++++++++++++++++++++++++++++

5100 Rem --- Schleife, Abtasten des Spielfeldes
5110 For I=1 to 5: For J=1 to 5: B(J,I)=A(J,I): Next J: Next I
5120 MAX=-100
5130 For VY=2 to 5: For VX=1 to 5
5140 If A(VX,VY)<>2 then 5200
5150 If A(VX,VY-1)=0 then NX=VX: NY=VY-1: Gosub 7000
5160 If VX=1 then 5180
5170 If A(VX-1,VY-1)=1 then NX=VX-1: NY=VY-1: Gosub 7000
5180 If VX=5 then 5200
5190 If A(VX+1,VY-1)=1 then NX=VX+1: NY=VY-1: Gosub 7000
5200 Next VX: Next VY

5300 Rem --- Zug ausfuehren und ausgeben
5310 XV=Z1: YV=Z2: XN=Z3: YN=Z4: Gosub 3000
5320 A(XV,YV)=0: A(XN,YN)=2
5330 Print "Mein Zug: Von ";XV;",";YV;" nach ";XN;",";YN
5340 Return

6000 Rem +++ Spielende? ++++++++++++++++++++++++++++++++++++++++
6010 AUS=0
```

```
6100 Rem --- Hat Computer gewonnen?
6110 For I=1 to 5
6120 If A(I,1)=2 then AUS=1: Return
6130 Next I
6140 For I=1 to 5: For J=1 to 5
6150 If A(J,I)=1 then 6200
6160 Next J: Next I
6170 AUS=1: Return

6200 Rem --- Hat Spieler gewonnen?
6210 For I=2 to 5: For J=1 to 5
6220 If A(J,I)<>2 then 6280
6230 If A(J,I-1)=0 then Return
6240 If J=1 then 6260
6250 If A(J-1,I-1)=1 then Return
6260 If J=5 then 6280
6270 If A(J+1,I-1)=1 then Return
6280 Next J: Next I
6290 AUS=1: Return

7000 Rem +++ Bewertung ++++++++++++++++++++++++++++++++++++++++++++

7100 Rem --- Zug ausfuehren, Spielfeld abtasten
7110 A(VX,VY)=0: A(NX,NY)=2: BEW=0
7120 For Y=1 to 5: For X=1 to 5
7130 If A(X,Y)=1 then gosub 7300
7140 If A(X,Y)=2 then gosub 7400
7150 Next X: Next Y

7200 Rem --- Spielfeld neu setzen, BEW>MAX?
7210 For I=1 to 5: For J=1 to 5: A(J,I)=B(J,I): Next J: Next I
7220 If BEW<MAX or (BEW=MAX and Rnd(0)>.4) then Return
7230 MAX=BEW: Z1=VX: Z2=VY: Z3=NX: Z4=NY
7240 Return

7300 Rem --- Feld mit Springer
7310 BEW=BEW-10
7320 For I=1 to 8
7330 If X+X(I)<1 or X+X(I)>5 or Y+Y(I)<1 or Y+Y(I)>5 then 7380
7340 If A(X+X(I),Y+Y(I))<>2 then 7380
7350 BEW=BEW-5
```

```
7355 If Y+Y(I)=5 then 7380
7360 If X+X(I)=1 then 7370
7365 If A(X+X(I)-1,Y+Y(I)+1)=2 then BEW=BEW+2
7370 If X+X(I)=5 then 7380
7375 If A(X+X(I)+1,Y+Y(I)+1)=2 then BEW=BEW+2
7380 Next I
7390 Return

7400 Rem --- Feld mit Bauer
7410 If Y=1 then BEW=500: Return
7420 If A(X,Y-1)=0 then BEW=BEW+1
7430 If X=1 then 7450
7440 If A(X-1,Y-1)=1 then BEW=BEW+2
7450 If X=5 then 7470
7460 If A(X+1,Y-1)=1 then BEW=BEW+2
7470 Return

8000 Rem +++ Datas Springerzuege ++++++++++++++++++++++++++++++++
8010 Data -2,1,-1,2,1,2,2,1,-2,-1,-1,-2,1,-2,2,-1
```

DRITTER TEIL

DER COMPUTER LERNT

KAPITEL 6: BAUERNKAMPF

```
-+-+-+-
 !S! !

Ihr Zug nach Feld? 7

C! !C!
-+-+-+-
 ! ! !
-+-+-+-
 ! !S!
-+-+-+-
 ! ! !

Mein Zug: 15 nach 10

C! ! !
-+-+-+-
 !C! !
-+-+-+-
 ! !S!
-+-+-+-
 ! ! !

Ihr Zug nach Feld ?12
```

EINFÜHRUNG

Computer tun normalerweise nur genau das, was man ihnen eingegeben hat. Ihr Verhalten ist vom Programmierer exakt festgelegt und bis ins kleinste Detail vorausbestimmbar. Der Computer hat hierbei überhaupt keinen eigenen Willen. Er ist lediglich ein sehr schnell und sehr zuverlässig arbeitender Sklave des Menschen.

Der erste Schritt weg von diesem Prinzip erfolgt durch den Einbau von Zufallsgeneratoren. Diese Zufalls-Generatoren liefern, wie der Name schon sagt, zufällige Werte. Jeder Wert löst dabei eine ganz bestimmte Reaktion aus. Da diese Werte zufällig sind, ist das Verhalten der Maschine nicht mehr genau voraussagbar.

Der zweite Schritt besteht darin, dem Computer die Möglichkeit zu geben, durch Erfahrung zu lernen. Wie das im einzelnen geschieht, werden Sie auf den nächsten Seiten ausführlich erfahren. Diese lernenden Computer bilden den Grundstein für eine immer größer werdende Furcht. Viele befürchten, daß wir eines Tages vieleicht die Kontrolle über diese lernenden Maschinen verlieren könnten. Diese Maschinen könnten gleichsam einen eigenen Willen entwickeln. Diese Furcht ist gewiss nicht unbegründet. Ein Computer, entsprechend programmiert, könnte durch Erfahrung schnell einen Punkt erreichen, der weit über dem Niveau seines Programmierers liegt und deshalb von diesem nicht mehr verstanden werden könnte. Hier kann es dann schnell passieren, daß nicht mehr der Computer der Sklave ist, sondern der Mensch der Sklave des Computers.

DIE LERNMETHODE NACH DONALD MITCHIE

Donald Mitchie ist Biologe an der Universität Edinburg. Von ihm stammt die Idee, aus Streichholzschachteln und farbigen Perlen eine einfache Lernmaschine zu konstruieren. Dabei geht man folgendermaßen vor:

Jede mögliche Spielstellung wird auf einer Streichholzschachtel aufgezeichnet. Wenn genau festgelegt ist, wer anfängt, so werden nur die Spielstellungen benötigt, denen die Maschine gegenübersteht. Fängt jedoch einmal der Spieler und einmal die Maschine an, so müssen alle Spielstellungen gezeichnet werden. Auf diese Zeichnungen markiert man dann mit verschiedenen farbigen Pfeilen alle möglichen Züge der Maschine. In das Innere der Schachteln wird für jeden Pfeil eine gleichfarbige Perle gelegt. Zu Anfang befinden sich also in jeder Schachtel genauso viele Perlen wie mögliche Züge der Maschine. Hat man dies erledigt, so kann das Spiel beginnen.

Jedesmal, wenn die Maschine am Zug ist, sucht man diejenige Streichholzschachtel, welche die gerade vorhandene Spielstellung wiedergibt. Dann zieht man blind aus dieser Schachtel eine Perle heraus und legt diese auf den Deckel der Schachtel. Der Zug mit der Farbe dieser Perle wird anschließend ausgeführt.

Dies geht solange, bis das Spiel entschieden ist. Hat die Maschine gewonnen oder liegt ein Unentschieden vor, so werden alle Perlen wieder unverändert in die Schachteln zurückgelegt und das Spiel beginnt von neuem. In diesem Falle lernt die Maschine also nicht.

Verliert sie aber, so straft man sie durch Wegnehmen der Perle des letzten Zuges. Auf diese Weise wird die Maschine auf Dauer lernen, diejenigen Züge zu bevorzugen, welche zu einem Gewinn oder zu einem Unentschieden führen.

Entsprechend wird sie die Züge, die zum Verlust des Spieles führen, langsam fallenlassen. Gelangt man im Laufe eines Spieles zu einer Schachtel, welche keine Perle mehr enthält, so heißt das, daß alle Züge der Maschine bei richtigem Weiterspielen des Gegners unweigerlich zu einer Niederlage führen. In diesem Falle gibt die Maschine auf und man entfernt wieder die letzte Perle.

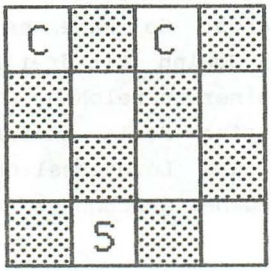
Das Programm, welches Ihnen auf den nächsten Seiten vorgestellt wird, arbeitet nach dieser Lernmethode. Zu Beginn des Spieles kennt der Computer lediglich die Spielregeln, in dieser Phase ist er leicht zu schlagen. Je mehr Spiele er aber verliert, desto besser spielt er. Nach genau 33 verlorenen Spielen ist er unschlagbar geworden.

Man kann die Maschine übrigens auch auf andere Weise belohnen und bestrafen. So wäre es z.b. möglich, die Maschine bei einem Gewinn mit drei Perlen und bei einem Unentschieden mit einer zu belohnen. Bei einer Niederlage könnte wie gehabt eine Perle entfernt werden. Es wäre einmal interessant, das Lernverhalten der Maschine in Abhängigkeit verschiedener Belohnungen zu untersuchen.

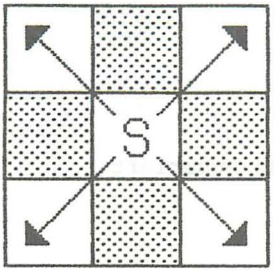
BAUERNKAMPF

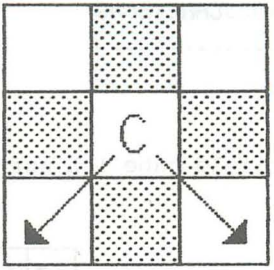
Die gerade beschriebene Lernmethode will ich nun an einem einfachen Spiel anwenden, welchem ich den Namen "Bauernkampf" gegeben habe.

Das Spielfeld besteht aus einem 4 X 4-Spielbrett, wobei nur auf den weißen Feldern gespielt wird. Der Computer besitzt zwei mit "C" gekennzeichnete Bauern, welche sich am oberen Rand des Spielfeldes befinden. Der Spieler dagegen besitzt nur einen mit "S" gekennzeichneten Bauern, welcher am linken unteren Rand startet.

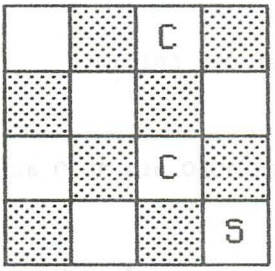


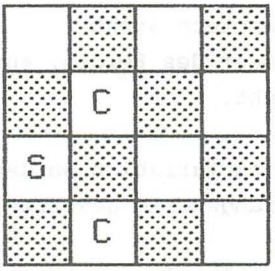
Der Spieler macht den ersten Zug, danach wird abwechselnd gezogen. Die Bauern dürfen sich diagonal um ein Feld auf ein leeres angrenzendes Feld bewegen. Die Bauern des Computers dürfen sich dabei aber nur in Richtung des unteren Spielfeldrandes bewegen, also nicht nach oben. Die möglichen Züge des Spielers und des Computers sind demzufolge also:





Die Aufgabe des Spielers besteht darin, mit seinem Bauern den oberen Spielfeldrand zu erreichen. Der Computer muß nun versuchen, den Spieler an diesem Vorhaben zu hindern. Schafft es der Computer, den Bauern des Spielers so zu blockieren, daß dieser zugunfähig geworden ist, so hat der Computer das Spiel gewonnen. Mögliche Gewinnstellungen des Computers sind also zum Beispiel (der Spieler ist am Zug):





PROGRAMMBESCHREIBUNG

Das Spielfeld habe ich auf folgende Weise numeriert:

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Eine Spielstellung wird nun durch drei Variablen festgehalten:

C1 = Nummer des Feldes, auf dem der erste Bauer des Computers steht

C2 = Nummer des Feldes, auf dem der zweite Bauer des Computers steht.

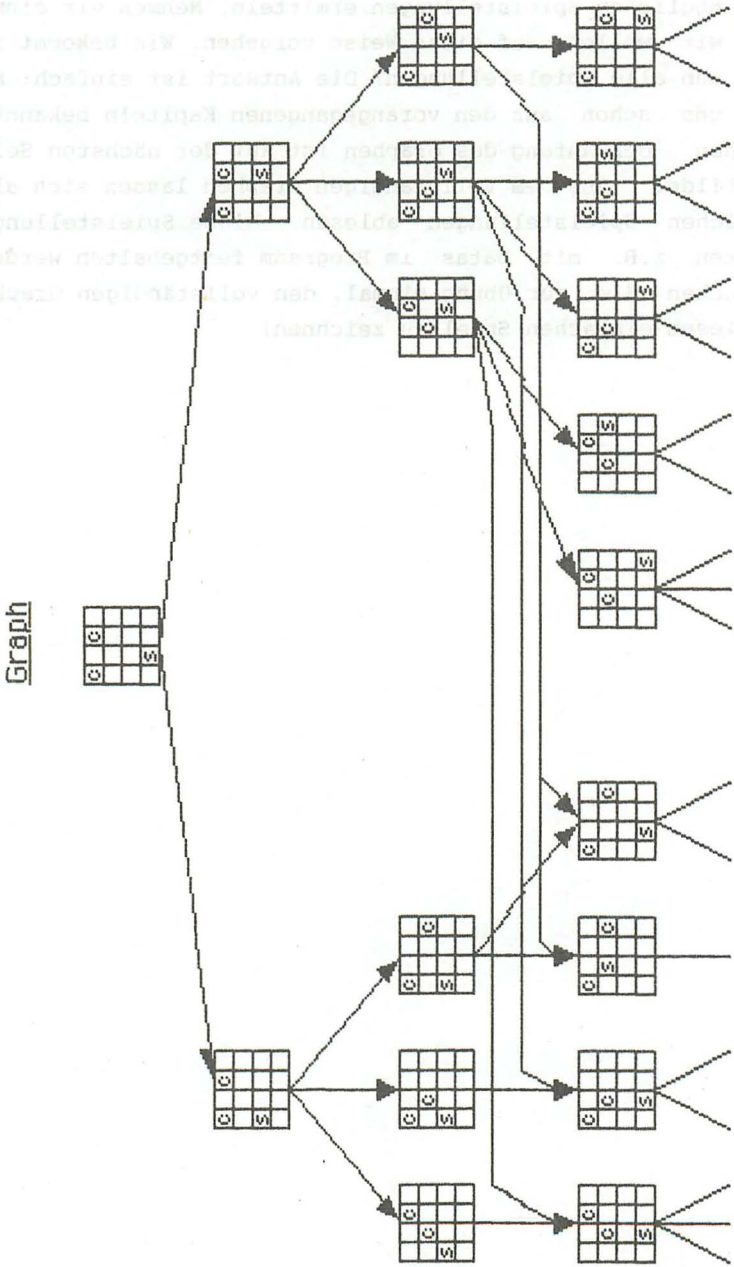
SP = Nummer des Feldes, auf dem der Bauer des Spielers steht.

Diese drei Variablen werden in den Zeilen 1050 bis 1070 auf ihre Anfangswerte gesetzt.

Eine Spielstellung könnte natürlich auch auf andere Weise festgehalten werden. Es wäre z.B. möglich, für das Spielfeld eine Tabelle A(4,4) anzulegen. Jedoch würde diese Methode erheblich mehr Speicherplatz benötigen. Es ist die Aufgabe des Programmierers, die günstigste Form auszuwählen.

Die Variable F wird Sie vieleicht etwas verwundern. Was heißt das, die Anzahl der bekannten Spielfelder? Um das zu verstehen, muß ich etwas weiter ausholen.

Nach der Lernmethode von Donald Mitchie müßte man zuerst alle möglichen Spielstellungen ermitteln. Nehmen wir einmal an, wir wollen auf diese Weise vorgehen. Wie bekommt man aber nun alle Spielstellungen? Die Antwort ist einfach: Mit dem uns schon aus den vorangegangenen Kapiteln bekannten Graphen. Der Anfang des Graphen ist auf der nächsten Seite abgebildet. Aus dem vollständigen Graphen lassen sich alle möglichen Spielstellungen ablesen. Diese Spielstellungen könnten z.B. mit Datas im Programm festgehalten werden. Versuchen Sie zur Übung einmal, den vollständigen Graphen zu diesem einfachen Spiel zu zeichnen!


Graph

Haben Sie den Graphen vollständig gezeichnet oder haben Sie vorher aufgegeben? In beiden Fällen jedenfalls haben Sie bemerkt, wie umfangreich und aufwendig so ein Graph schon bei einfachen Spielen werden kann. Es ist daher ratsam, nach folgender Methode vorzugehen:

Betrachten wir dazu wieder die Lern-Maschine aus Zündholzschachteln. Es ist überhaupt nicht nötig, zu Beginn alle Spielstellungen aufzuzeichnen. Viel einfacher ist es, die Spielstellungen erst ***WÄHREND*** des Spieles zu zeichnen. Immer wenn eine Spielstellung vorliegt, die wir noch nicht aufgezeichnet haben, wird eine neue Streichholzschachtel mit dieser Stellung angefertigt. Auf diese Weise erhalten wir allmählich alle möglichen Spielstellungen.

Der Vorteil dieser Methode in Bezug zu unserem Computerspiel leuchtet uns nun sofort ein. Wir können den Computer nämlich so programmieren, daß er jede ihm unbekannte Spielstellung ***SELBSTÄNDIG*** festhält. Dadurch sparen wir uns die Mühe, alle Spielstellungen aufzusuchen und im Programm (eventuell mit Datas) abzuspeichern.

Die Variable F bereitet uns jetzt keine Schwierigkeiten mehr. Die Variable F gibt also an, wieviele verschiedene Spielstellungen der Computer inzwischen festgehalten hat. Da er ganz zu Beginn des Spieles überhaupt keine Spielstellung kennt, wird F in Zeile 1040 mit Null initialisiert.

Die Variable G wird für die Anzeige des Gewinners verwendet. Bei G=1 hat der Spieler gewonnen, bei G=2 der Computer. Betrachten Sie dazu das in Zeile 6000 beginnende Unterprogramm "Spielende".

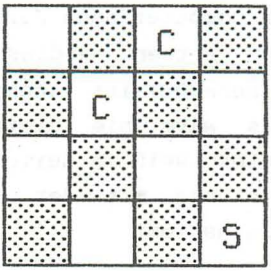
Das Programm ist so gestaltet, daß es leicht verstanden

werden kann. Interessant ist natürlich vor allem der "Computerzug". Ihn wollen wir jetzt näher betrachten.

Die Spielstellungen werden in der Tabelle F(50,3) festgehalten. Dabei geben F(N,1) und F(N,2) die Positionen der beiden Bauern des Computers an; F(N,3) bezeichnet die Position des Bauern des Spielers. In den Zeilen 5010-5060 schaut der Computer nun nach, ob die ihm vorliegende Spielstellung schon bekannt ist. Dazu wird in Zeile 5030 eine Schleife T gestartet, welche von 1 bis F zählt.
Bei jedem Schleifendurchgang wird die vorliegende Spielstellung mit der Stellung F(T,1), F(T,2) und F(T,3) verglichen. Sind beide Stellungen gleich, so springt der Computer nach Zeile 5100. Wurde die Schleife aber erfolglos durchlaufen, ist der Computer also in Zeile 5060 angelangt, so verzweigt er nach Zeile 5500. In diesem Falle ist die Spielstellung also noch nicht bekannt.

Nehmen wir einmal an, die vorliegende Spielstellung ist nicht bekannt. Dann springt der Computer also nach Zeile 5500. Zunächst wird dann die Variable F incrementiert (um eins erhöht). Anschließend wird die Spielstellung abgespeichert. Das entspricht dem Zeichnen der Spielstellung auf eine neue Streichholzschachtel. Nun müssen noch sämtliche möglichen Züge des Computers ermittelt werden. Diese Züge werden in den Tabellen V(50,4) und N(50,4) festgehalten. Der Computer hat maximal vier Züge zur Auswahl, da jeder Bauer des Computers höchstens zwei Züge machen kann. Ist V(F,I) ungleich Null, so bedeutet das, daß der Computer seinen Bauern von Feld V(F,I) auf Feld N(F,I) ziehen kann.

Wie der Computer nun seine möglichen Züge ermittelt, will ich an einem Beispiel zeigen. Folgende gerade vorhandene Spielstellung sei also unbekannt:



Dann wird diese Stellung folgendermaßen abgespeichert (Zeile 5520):

F(F,1)=10, F(F,2)=15, F(F,3)=4

Der Computer testet nun, ob der erste Bauer nach links unten ziehen kann. Dies ist nicht der Fall, wenn dieser sich in der linken Spalte oder aber in der unteren Reihe befindet.

Die Felder in der linken Spalte sind dadurch gekennzeichnet, daß ihre Nummer um eins vermindert ohne Rest durch vier teilbar ist. In unserem Falle ist C1-1 nicht durch vier ohne Rest teilbar; der Bauer befindet sich demnach nicht in der linken Spalte. Der Bauer befindet sich nicht in der unteren Reihe, wenn die Nummer seines Feldes größer als 4 ist. Die Nummer des Feldes, das der Bauer bei einem Zug von Feld C1 nach links unten erreichen würde, ist C1-5. Dieses Feld muß natürlich leer sein, auf ihm darf sich weder der zweite Bauer des Computers noch der Bauer des Spielers befinden. Sind alle diese Tests erfolgreich durchgeführt worden, so ist der Zug von C1 nach C1-5 also möglich. V(F,1) wird deshalb mit C1 und N(F,1) mit C1-5 initialisiert.

Anschließend werden auf gleiche Weise der Zug des zweiten Bauern nach links vorne und die Züge nach rechts vorne untersucht. Zu unserem Beispiel erhält der Computer folgendes:

V(F,1) = 10	N(F,1) = 5
V(F,2) = 10	N(F,2) = 7
V(F,3) = 0	N(F,3) ist unwichtig, da der Zug nach links vorne nicht zulässig ist
V(F,4) = 15	N(F,4) = 12

Danach springt der Computer nach Zeile 5110, jedoch nicht ohne vorher T=F zu setzen. Da diese Spielstellung jetzt abgespeichert (also bekannt) ist, können wir nun einfach so weiterverfahren, als wäre sie gleich zu Beginn bekannt gewesen. Der Computer weiß in Zeile 5110 ja nur, daß die vorliegende Spielstellung mit der Stellung Nr. T in der Tabelle F(50,4) übereinstimmt.

Wir gelangen in jedem Falle also nach Zeile 5110. Dort wird nun getestet, ob der Computer überhaupt einen Zug zur Verfügung hat. Ist V(T,1) bis V(T,4) gleich Null, so ist kein Zug möglich, der nicht bei richtigem Weiterspielen des Spielers zum Verlust des Spieles führen würde. Diese Situation entspricht dem Fall "Keine Perle mehr in der Schachtel" bei unserer Zündholz-Lernmaschine. Ist die Summe also gleich Null, so gibt der Computer deshalb auf.

Ist die Summe ungleich Null, so ist mindestens ein Zug noch zulässig. In den Zeilen 5120 und 5130 wird ein zulässiger Zug nun zufällig ausgewählt. Es ist der Zug von Feld V(T,R) nach Feld N(T,R). Dieser Zug wird in D1 und D2 festgehalten. D1 und D2 geben also immer den zuletzt gemachten Zug an. Diesen Zug müssen wir ja kennen.

Ist der Zug ausgeführt und abgespeichert, so muß anschließend nachgeprüft werden, ob das Spiel zu Ende ist, d.h. ob der Spieler durch den letzten Zug zugunfähig geworden ist. Dies geschieht in den Zeilen 5200-5280. Ist der Test negativ, d.h. ist das Spiel noch nicht zu Ende, so erfolgt ein Return zum Hauptprogramm. Ansonsten gelangen wir zum in Zeile 6000 beginnenden Unterprogramm "Spielende". Dort wird zunächst das Spielfeld, dann der Gewinner ausgegeben. Nur wenn der Computer verloren hat (G=1) tritt der Lerneffekt in Kraft. Wie wir wissen, muß die Perle des letzten Zuges entfernt werden. Dies geschieht durch V(D1,D2)=0.

Versuchen Sie zur Übung einmal, das Programm so zu erweitern, daß sowohl der Spieler als auch der Computer anfangen kann, wobei dem Spieler die Wahl überlassen bleibt!

Wie bereits erwähnt, spielt der Computer nach 33 verlorenen Spielen perfekt. In Anbetracht dieses doch wirklich einfachen Spieles ist das eine enorm hohe Zahl. Im nächsten Kapitel werden wir deshalb Methoden behandeln, welche das Lernen des Computers wesentlich beschleunigen!!

Programmlauf

++ Bauernkampf ++

```
 C!  !C !
-+-+-+-+-
 !  !  !
-+-+-+-+-
 !  !  !
-+-+-+-+-
 !S!  !
```

Ihr Zug nach Feld ?5

```
 C!  !C !
-+-+-+-+-
 !  !  !
-+-+-+-+-
S!  !  !
-+-+-+-+-
 !  !  !
```

Mein Zug: 13 nach 10

```
 !  !C !
-+-+-+-+-
 !C!  !
-+-+-+-+-
S !  !  !
-+-+-+-+-
 !  !  !
```

Ihr Zug nach Feld ?2

```
 !  !C !
-+-+-+-+-
 !C!  !
-+-+-+-+-
 !  !  !
-+-+-+-+-
 !S!  !
```

Mein Zug: 10 nach 7

```
 !  !C !
-+-+-+-+-
 !  !  !
-+-+-+-+-
 !  !C !
-+-+-+-+-
 !S!  !
```

Ihr Zug nach Feld ?5

```
 !  ! C!
-+-+-+-+-
 !  !  !
-+-+-+-+-
S!  ! C!
-+-+-+-+-
 !  !  !
```

Mein Zug: 15 nach 10

```
 !  !  !
-+-+-+-+-
 !C!  !
-+-+-+-+-
S!  !C !
-+-+-+-+-
 !  !  !
```

Ihr Zug nach Feld ?2

```
 !  !  !
-+-+-+-+-
 !C!  !
-+-+-+-+-
 !  ! C!
-+-+-+-+-
 !S!  !
```

Mein Zug: 10 nach 5

```
 !  !  !
-+-+-+-+-
 !  !  !
-+-+-+-+-
C!  ! C!
-+-+-+-+-
 !S!  !
```

Ich habe gewonnen

VARIABLENLISTE

F(X,Y)	:Speicherung der bekannten Spielfelder
V(X,Y), N(X,Y)	:Speicherung der Züge des Computers zu den bekannten Spielfelden
G	:Indikator für Gewinner
F	:Anzahl der bekannten Spielfelder
C1,C2	:Positionen der Bauern des Computers
SP	:Positionen des Bauern des Spielers
D1,D2	:Abspeichern des letzten Zuges des Computers

Listing: Bauernkampf

```
1000 Rem +++ Initialisierung ++++++++++++++++++++++++++++++++++++
1010 Dim F(50,3)          :Rem Bekannte Spielfelder
1020 Dim V(50,4),N(50,4)  :Rem Zuege der Spielfelder
1030 G=0                  :Rem Indikator fuer den Gewinner
1040 F=0                  :Rem Anzahl der bekannten Spielfelder
1050 C1=13                :Rem Position von Bauer 1 des Computers
1060 C2=15                :Rem Position von Bauer 2 des Computers
1070 SP=2                 :Rem Position von Bauer des Spielers

2000 Rem +++ Hauptprogramm ++++++++++++++++++++++++++++++++++++
2010 Graphics 0: Setcolor 2,2,2
2020 Print "++ Bauernkampf ++"
2030 Gosub 3000           :Rem Spielfeld ausgeben
2040 Gosub 4000           :Rem Spielerzug
2050 Gosub 3000           :Rem Spielfeld ausgeben
2060 Gosub 5000           :Rem Computerzug
2070 Goto 2030

3000 Rem +++ Spielfeld ausgeben ++++++++++++++++++++++++++++++++
3005 Print
3010 For I=4 to 1 Step -1
3020 For J=1 to 4
3030 X=4*(I-1)+J
3040 If C1=X or C2=X then print "C";: Goto 3070
3050 If SP=X then print "S";: Goto 3070
3060 Print " ";
3070 If J<4 then print "!";
3080 Next J
3090 Print
3100 If I>1 then print "-+-+-+-"
3110 Next I
3120 Print
3130 Return

4000 Rem +++ Spielerzug ++++++++++++++++++++++++++++++++++++++++
4010 Rem --- Eingabe des Zuges
4020 Print "Ihr Zug nach Feld ";
4030 Input X
```

```
4100 Rem --- Ist Zug korrekt?
4110 If X<1 or X>16 or C1=X or C2=X then 4020
4120 D=X-SP
4130 If (D=5 or D=-3) and SP/4=Int(SP/4) then 4020
4140 If (D=3 or D=-5) and (SP-1)/4=Int((SP-1)/4) then 4020
4150 If Abs(D)<>3 and Abs(D)<>5 then 4020

4200 Rem --- Zug ist korrekt
4210 SP=X
4220 If SP=13 or SP=15 then G=1: Goto 6000
4230 Return

5000 Rem +++ Computerzug ++++++++++++++++++++++++++++++++++++++++

5010 Rem --- Ist Stellung bekannt?
5020 If F=0 then 5500          :Rem Stellung ist nicht bekannt
5030 For T=1 to F
5040 If SP=F(T,3) and ((C1=F(T,1) and C2=F(T,2)) or (C1=F(T,2)
     and C2=F(T,1))) then 5100
5050 Next T
5060 Goto 5500                 :Rem Stellung ist nicht bekannt

5100 Rem --- Stellung ist bekannt
5110 If V(T,1)+V(T,2)+V(T,3)+V(T,4)=0 then print "Ich gebe auf":
     G=1: Goto 6000
5120 R=Int(Rnd(0)*4)+1
5130 If V(T,R)=0 then 5120
5140 Print "Mein Zug: ";V(T,R);" nach ";N(T,R)
5150 If C1=V(T,R) then C1=N(T,R)
5160 If C2=V(T,R) then C2=N(T,R)
5170 D1=T: D2=R                :Rem Speicherung des letzten Zuges

5200 Rem --- Spielende?
5210 For I=-5 to 5 Step 2
5220 If I=-1 then I=3
5230 If SP+I<1 or SP+I>16 then 5270
5240 If (I=-5 or I=3) and (SP-1)/4=Int((SP-1)/4) then 5270
5250 If (I=-3 or I=5) and SP/4=Int(SP/4) then 5270
5260 If C1<>SP+I and C2<>SP+I then Return
5270 Next I
5280 G=2: Goto 6000
```

```
5500 Rem --- Unbekannte Stellung
5510 F=F+1
5520 F(F,1)=C1: F(F,2)=C2: F(F,3)=SP
5530 For I=1 to 4: V(F,I)=0: Next I
5540 If (C1-1)/4<>Int((C1-1)/4) and C1>5 and C1-5<>SP and C1-5<>
     C2 then V(F,1)=C1: N(F,1)=C1-5
5550 If (C2-1)/4<>Int((C2-1)/4) and C2>5 and C2-5<>SP and C2-5<>
     C1 then V(F,3)=C2: N(F,3)=C2-5
5560 If C1/4<>Int(C1/4) and C1>3 and C1-3<>SP and C1-3<>C2 then
     V(F,2)=C1: N(F,2)=C1-3
5570 If C2/4<>Int(C2/4) and C2>3 and C2-3<>SP and C2-3<>C1 then
     V(F,4)=C2: N(F,4)=C2-3
5580 T=F: Goto 5110

6000 Rem +++ Spielende ++++++++++++++++++++++++++++++++++++++++++
6010 Gosub 3000                    :Rem Spielfeld
6020 Print
6030 If G=1 then print "Sie haben gewonnen!"
6040 If G=2 then print "Ich habe gewonnen"
6050 Print
6060 C1=13: C2=15: SP=2
6070 If G=1 then V(D1,D2)=0        :Rem Computer lernt
6080 For I=1 to 1000: Next I
6090 Goto 2000                     :Rem Neues Spiel
```

KAPITEL 9: VERBESSERUNG DER LERNMETHODE

WAS WOLLEN WIR ÜBERHAUPT VERBESSERN?

Zunächst müssen wir uns natürlich darüber im klaren sein, was wir überhaupt verbessern wollen. Welche Mängel haben wir zu beanstanden? Diese Frage wurde bereits mehrfach angeschnitten und zum Teil auch schon beantwortet. Zwei Mängel hat unsere Lernmethode:

Der erste besteht darin, daß wir alle möglichen Spielstellungen abspeichern müssen. Dadurch eignet sich die bisherige Methode nur für sehr einfache Spiele, welche wenige Spielstellungen aufweisen. Um auch etwas kompliziertere Spiele programmieren zu können, muß die Anzahl der abzuspeichernden Spielstellungen reduziert werden. Das ist unsere erste Aufgabe.

Der zweite Mangel ist die Tatsache, daß der Computer nur sehr langsam lernt. Das liegt zum einen an der hohen Anzahl der Spielstellungen und zum anderen daran, daß der Computer seine Züge völlig zufällig auswählt. Unsere zweite Aufgabe besteht also darin, das Lernen des Computers zu beschleunigen.

Es folgt nun eine Erläuterung der wichtigsten Methoden zur Verbesserung der Lernmethode. Danach sind wir in der Lage, die Lernmethode auf viele Spiele wirkungsvoll anzuwenden.

CPMPUTER GEGEN SICH SELBST SPIELEN LASSEN

Bei komplexen Spielen dauert es sehr lange, bis der Computer gelernt hat, stark zu spielen. Hat man dies endlich einmal geschafft, so soll diese Spielstärke wenn möglich auch an anderen Spieltagen wieder verfügbar sein. Um dies zu erreichen, müssen die Spielstellungen und die Tabellen der zugehörigen Züge auf ein geeignetes Medium (Diskette oder Tape) abgespeichert werden. Das Programm muß dann so gestaltet sein, daß es beim ersten "Start" diese Daten von dem Medium einliest. Auf diese Weise würde das Programm immer stärker werden, bis es schließlich optimal spielen würde.

Was ist aber, wenn wir gleich zu Beginn und nicht erst nach mühevollem tage- oder sogar wochenlangem Spielen ein spielstarkes Programm wollen? Hier bietet sich nun die Möglichkeit an, den Computer gegen sich selbst spielen zu lassen. Wir müssen das Programm so auslegen, daß der Computer zu Beginn vieleicht 100, 1000 oder sogar noch mehr Spiele gegen sich selber spielt. Danach werden die Daten, wie im obigen Abschnitt erklärt, abgespeichert. Bei einer genügend hohen Anzahl von Spielen steht uns nun die gewünschte Spielstärke zur Verfügung.

Die Sache hat natürlich einen Haken. "1000 Spiele gegen sich selbst spielen, das dauert ja ewig in Basic" werden Sie vieleicht sagen. Optimal wäre hier z.B. ein Basic-Compiler, welcher die nötige Zeit erheblich herabsetzen würde. Aber auch ohne Basic-Compiler läßt sich diese Methode anwenden. Es ist immer noch besser, den Computer vieleicht eine Nacht lang gegen sich selbst spielen zu lassen, als diese ganze Nacht lang selbst gegen den Computer spielen zu müssen. Zur Not kann man diese Zeit auch auf mehrere Tage ausweiten, das Ganze muß ja nur einmal geschehen!

SYMMETRIE UND ABSTRAKTION

Die große Anzahl der verschiedenen Spielstellungen ist das Hauptproblem der Lernmethode. Die Grenze des Speicherplatzes ist schnell erreicht. Hier muß nun Abhilfe geschafft werden.

Betrachten wir dazu das bekannte Spiel "Vier gewinnt":

Das Spielfeld besteht aus sieben Spalten und sechs Reihen, wobei die Spalten von links nach rechts mit 1 bis 7 numeriert sind. Zwei Spieler mit unterschiedlich gekennzeichneten Spielsteinen sind abwechselnd am Zug. Ein Zug besteht darin, einen Spielstein auf das Spielfeld zu setzen. Die Spalten dürfen jedoch nur von unten aufgefüllt werden. Das Ziel ist, vier eigene Steine waagrecht oder diagonal in einer Reihe anzuordnen.

Schauen Sie sich nun die zwei folgenden Spielstellungen an, wobei der Computer ("O") am Zug sein soll!

```
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! !X! ! ! !
+-+-+-+-+-+-+-+
! !X! !O! ! ! !
+-+-+-+-+-+-+-+
```

```
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! !X! ! ! !
+-+-+-+-+-+-+-+
! ! ! !O! !X! !
+-+-+-+-+-+-+-+
```

Normalerweise würde der Computer diese zwei Stellungen getrennt abspeichern. Das ist jedoch nicht nötig, da beide Stellungen symmetrisch sind. Dadurch, daß symmetrische Stellungen erkannt werden, verringert sich die Anzahl der abzuspeichernden Spielstellungen in der Regel beträchtlich.

Eine weitere Möglichkeit besteht darin, dem Computer ein gewisses Abstraktionsvermögen zu vermitteln. Der Computer muß die Spielstellungen auf die spielbestimmenden Elemente hin untersuchen und sie von den unwichtigen Details trennen.

Zwei sich nur unwesentlich unterscheidende Spielstellungen wären bei dem Spiel "Vier gewinnt" z.B.:

```
+-+-+-
```

ZÜGE, WELCHE ZU VERLUSTSTELLUNGEN FÜHREN,

FRÜHZEITIG ERKENNEN

Der Lerneffekt beruht wie wir wissen darauf, die Perlen derjenigen Züge zu entfernen, welche zu Verluststellungen führen. Hier bietet sich nun eine weitere Verbesserung an. Betrachten Sie dazu wieder die folgende Spielstellung, der Computer ("O") sei am Zuge:

```
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! ! ! ! ! !
+-+-+-+-+-+-+-+
! ! ! !O! ! ! !
+-+-+-+-+-+-+-+
! ! !X!X! ! ! !
+-+-+-+-+-+-+-+
 1 2 3 4 5 6 7
```

Der Computer hat sieben Züge zur Auswahl. Welchen Zug würden Sie ihm empfehlen? Nun, Sie haben sicher bemerkt, daß der Computer die Spalte 2 oder 5 auffüllen muß, wenn er nicht verlieren will. Alle anderen Züge führen zum Verlust des Spieles.

Der Computer weiß das natürlich nicht. Er würde zufällig einen Zug auswählen und diesen ausführen. Es würde ziemlich lange dauern, d.h. der Computer müßte erst viele Spiele verlieren, bis er gelernt hätte, immer Spalte 2 oder Spalte 5 aufzufüllen.

Sie haben sicher schon gemerkt, worauf ich hinaus will. Eine wirkungsvolle Verbesserung der Lernmethode bestünde darin, Züge zu Verluststellungen frühzeitig zu erkennen und die "Perlen" dieser Züge daraufhin zu entfernen. Dazu müßte der Computer alle seine möglichen Züge ausführen, daß heißt alle Züge, zu denen noch Perlen vorhanden sind, denn nur

solche kommen ja überhaupt in Betracht. Bei jedem Zug müßten dann alle Züge des Spielers ausgeführt werden usw. Je mehr solcher Halbzüge der Computer vorausschaut, desto schneller lernt er, desto länger natürlich aber auch die Denkzeit.

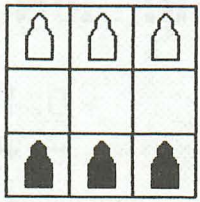
Bei unserem Beispiel müßte der Computer vier Halbzüge vorausschauen. Das heißt, er müßte seine Züge, bei jedem Zug die des Spielers, daraufhin wieder seine und schließlich nochmals die des Gegners ausführen. Erst dann würde er bemerken, daß alle Züge außer Spalte 2 und 5 zum Verlust des Spieles führen. Da jeder maximal sieben Züge zur Auswahl hat, wären das also höchstens 7^4=2401 auszuführende Züge.

Ist dies erst einmal geschafft, so ist es ein leichtes, den Computer nicht nur Züge zu Verluststellungen erkennen zu lassen, sondern auch Züge zu Gewinnstellungen. Das wäre eine weitere Verbesserung.

```
** HEXAPAWN **

1 ! 2 ! 3      0 ! 0 ! 0
- + - + -      - + - + -
4 ! 5 ! 6        !   !
- + - + -      - + - + -
7 ! 8 ! 9      X ! X ! X

Ihr Zug  von 9
Ihr Zug nach 6
```

SPIELREGELN

Hexapawn (Sechs-Bauern-Spiel) ist ein einfaches Spiel, welches von Martin Gardner in dem Buch "Logik unterm Galgen" beschrieben wird. Auf einem 3 X 3-Brett sind je drei Bauern auf folgende Weise angeordnet:



Die Bauern können geradeaus um ein Feld auf ein leeres Feld vorrücken. Außerdem kann ein gegnerischer Bauer geschlagen werden, indem ein Feld diagonal nach links oder rechts auf ein vom Gegner besetztes Feld gezogen wird. Der geschlagene Bauer wird dabei vom Brett entfernt. Spieler Weiß und Spieler Schwarz ziehen abwechselnd, wobei bei jedem Zug ein Bauer bewegt wird. Es gewinnt derjenige, welcher

-einen Bauern auf die gegnerische Seite bringt oder

-alle Bauern des Gegners schlägt oder

-die Bauern des Gegners so blockiert, daß dieser nicht mehr ziehen kann.

PROGRAMMERLÄUTERUNG

Das Programm beginnt in Zeile 1000 mit der "Initialisierung". Das Spielfeld wird in der Tabelle A(X)

festgehalten. Die Tabellen F(X,Y), V(X,Y) und N(X,Y) dienen wieder zum Abspeichern der Spielfelder bzw. der zugehörigen Züge. In den Zeilen 1200 bis 1250 wird das Spielfeld initialisiert. Die Bauern des Computers werden durch eine 1 gekennzeichnet, die Bauern des Spielers erhalten eine -1.

Das "Hauptprogramm" umfaßt die Ausgabe der Spielregeln, das Setzen des Bildschirms und den Spielablauf. Die Ausgabe der Spielregeln erfolgt im Grafikmodus 0 mit weißer Schrift auf schwarzem Hintergrund und orangefarbener Umrandung. In Zeile 2310 wird der Spieler dazu aufgefordert, eine beliebige Taste zu drücken. Die Abfrage der Tastatur erfolgt mittels der Speicherstelle 753. Diese ist Null, falls keine Taste gedrückt wurde. Beim Betätigen einer Taste wechselt der Inhalt dieser Speicherstelle auf drei. Der Computer wartet so lange, bis der Spieler eine Taste drückt (außer "break", da "break" den Programmablauf beendet). Ist dies geschehen, so wird Grafikmodus 1 ohne Textfenster eingeschaltet. Außerdem werden in Zeile 2420 bzw. Zeile 2450 die Variablen ANF und BILD gesetzt. Die Variable BILD gibt die Anfangsadresse des Bildschirmspeichers wieder. Interessant ist nun die Variable ANF. Sie enthält die Anfangsadresse der Display-List. Durch "Poke ANF+6,7" wird erreicht, daß Zeile 1 und Zeile 2 des Bildschirms zu einer einzigen Zeile im Grafikmodus 2 zusammengefaßt werden. Zwei verschiedene Schriftgrößen erscheinen dadurch auf dem Bildschirm: Die oberste Zeile erscheint im Grafikmodus 2, die weiteren Zeilen im Grafikmodus 1. In Zeile 2440 wird nun in diese oberste Zeile der Text "++HeXaPaWn++" geschrieben. Durch die Verwendung von Großschrift, Kleinschrift, inverser Groß- und Kleinschrift erscheint diese Überschrift in vier Farben. Die Farbe verteilen sich dabei wie folgt:

Großschrift:	Orange
Kleinschrift:	Grün
inverse Großschrift:	Blau
inverse Kleinschrift:	Rosa

Im"Spielablauf" werden in gewohnter Weise die Unterprogramme "Spielfeld zeichnen", "Spielerzug" und "Computerzug" nacheinander aufgerufen. Der Spieler macht dabei immer den ersten Zug. Ungewohnt sind die Zeilen 2520 bis 2540. Sie bewirken, daß nach dem Computerzug der untere Teil des Bildschirms, der für die Eingabe und Ausgabe der Zuge verwendet wird, gelöscht wird. Das Löschen geht dabei folgendermaßen vonstatten: Der gesamte untere Teil des Bildschirms wird Zeile für Zeile nach oben geschoben, bis der ganze Text verschwunden ist. Dieses Löschen des Bildschirms wird akustisch etwas untermalt

Das Spielfeld wird in den Zeilen 3000 bis 3120 gezeichnet. Zu beachten ist, daß zwei Spielfelder nebeneinander gezeichnet werden. Das linke Spielfeld zeigt die Nummerierung der verschiedenen Felder an, das rechte Spielfeld gibt die Spielstellung wieder.

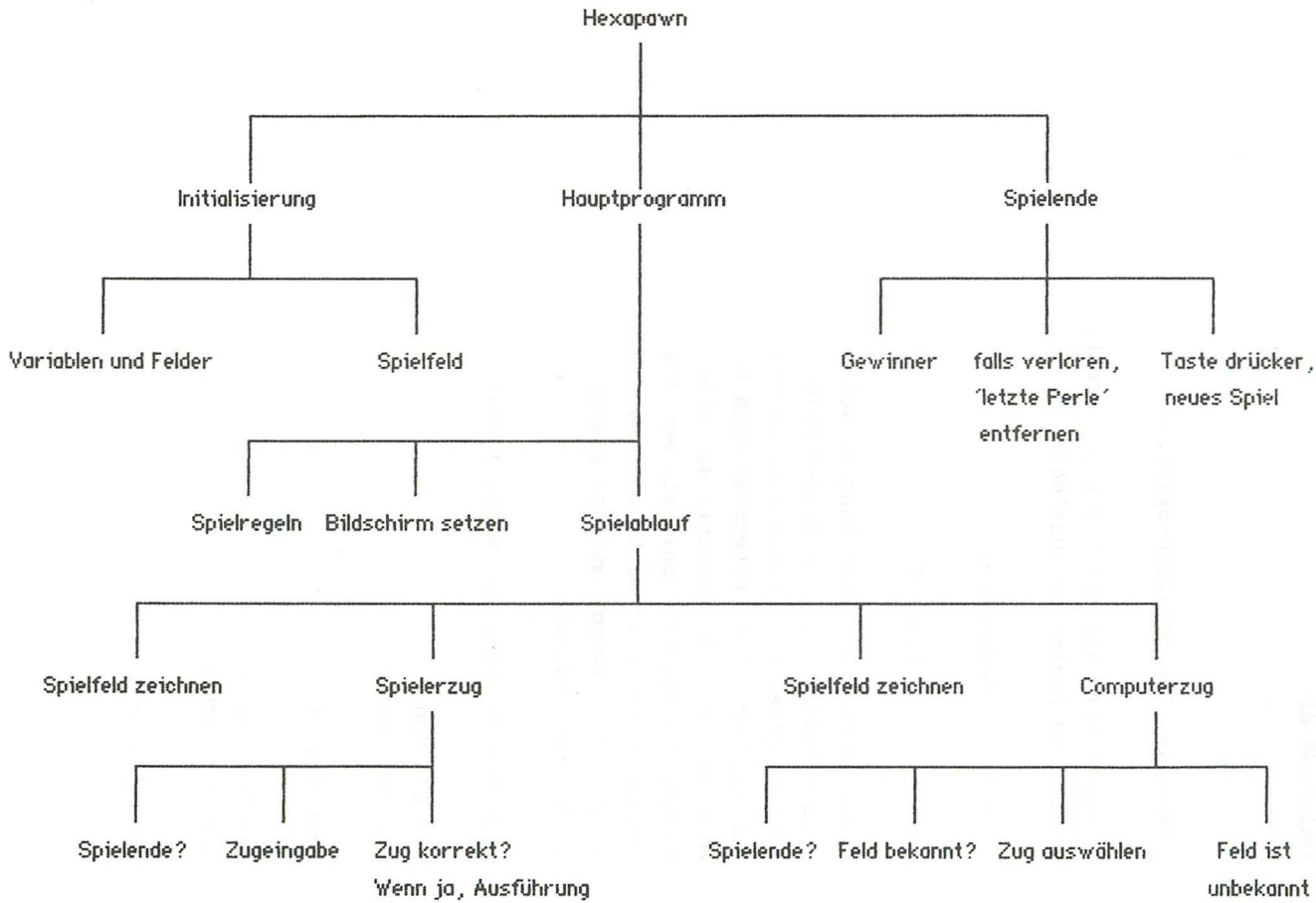
Der "Spielerzug" und der "Computerzug" beginnen beide mit dem Teil "Spielende?". Hierbei wird zunächst die Variable G als Indikator für den am Zug befindlichen Spieler gesetzt. Danach wird das in Zeile 6000 beginnende Unterprogramm "Alle möglichen Züge ermitteln" aufgerufen. Dieses Unterprogramm berechnet alle möglichen Züge und speichert diese in den Tabellen V(X,Y) und N(X,Y) ab. In Zeile 4120 bzw. Zeile 5120 wird nun geprüft, ob das Spiel zu Ende ist. Dies ist dann der Fall, wenn ein Bauer des Gegners die gegenüberliegende Seite erreicht hat oder wenn der am Zug befindliche Spieler nicht mehr ziehen kann. Fällt die Prüfung positiv aus, d.h. ist das Spiel zu Ende, so springt der Computer nach Zeile 7000. Dort erfolgt zunächst zur Unterstreichung des Spielendes eine akustische und optische Explosion. Danach wird der untere Teil des Bildschirms gelöscht und der Gewinner wird ausgegeben. Hat der Computer verloren, so wird außerdem der letzte Zug entfernt. Dies geschieht durch V(D1,D2)=0, wobei D1 die Nummer des Spielfeldes und D2 die Nummer des Zuges angibt. Anschließend muß der Spieler eine Taste drücken und das Spiel beginnt von neuem. Wichtig ist hierbei, daß das Spiel

nicht durch ein "Run" oder ein "Goto 1000" neu gestartet wir, da in diesem Falle ja alle Felder und Variablen wieder gelöscht würden. Die Felder F(X,Y), V(X,Y) und die Variable F müssen jedoch erhalten bleiben, wenn der Computer lernen soll. Während des Wartens auf das Drücken einer Taste wird die Kopfzeile des Bildschirms (Zeile mit doppelter Schrifthöhe und Text "Hexapawn") nach links geschoben. Der Text verschwindet dabei auf der linken Seite und taucht auf der rechten Seite wieder auf. Neu sind beim "Computerzug" die Variablen UNS und SYM. In den Zeilen 5200 bis 5330 wird getestet, ob die vorliegende Stellung schon bekannt ist. Zwei zueinander symmetrische Stellungen werden erkannt und nicht getrennt abgespeichert. Ist in Zeile 5300 UNS gleich 9, so ist die vorliegende Stellung identisch mit der Stellung T. Ist SYM gleich 9, so ist die vorliegende Stellung mit der Stellung T symmetrisch. S=0 heißt, daß die beiden Stellungen gleich sind, S=1 bedeutet beide Stellungen sind zueinander symmetrisch. Ist F gleich Null oder wurden alle Stellungen erfolglos geprüft, so wird in Zeile 5210 bzw. in Zeile 5330 nach Zeile 5500 gesprungen. In diesem Falle ist die vorliegende Stellung also noch nicht bekannt. Es wird nun F um 1 erhöht, anschließend wird die neue Stellung abgespeichert. Die Züge zu dieser Stellung wurden schon vom Unterprogramm "Alle möglichen Züge ermitteln" abgespeichert.

In Zeile 5420 wird ein Zug zufällig ausgewählt. Nun muß jedoch die Symmetrie beachtet werden. Ist S=1, so muß der Zug entsprechend zur symmetrischen Stellung geändert werden. Hierzu dienen die Variablen V1, V2, N1 und N2. Nach der Zugausgabe erfolgt in Zeile 5470 die Ausführung des Zuges. Die Variablen D1 und D2 dienen wieder wie bereits erwähnt zur Speicherung des letzten Zuges.

Hexapawn ist ein sehr einfaches Spiel. Dadurch, daß symmetrische Stellungen erkannt werden und damit nicht getrennt gespeichert werden, müssen maximal nur 19 verschiedene Stellungen abgespeichert werden.

Wie jedes Spiel, bei dem der Computer lernt, gilt auch hier, je besser der Gegner spielt, desto schneller lernt der Computer. Spielt der Gegner optimal, d.h. werden vom Gegner sinnlose Züge unterlassen, die den Computer auf überflüssige Seitenpfade lenken, so spielt der Computer nach acht verlorenen Spielen perfekt, d.h. nacht acht verlorenen Spielen ist er unschlagbar geworden. Derjenige Spieler, welcher den zweiten Zug macht, kann bei Hexapawn also immer gewinnen.


Hexapawn
Initialisierung
Hauptprogramm
Spielende
Variablen und Felder
Spielfeld
Gewinner
falls verloren, 'letzte Perle' entfernen
Taste drücker, neues Spiel
Spielregeln
Bildschirm setzen
Spielablauf
Spielfeld zeichnen
Spielerzug
Spielfeld zeichnen
Computerzug
Spielende?
Zugeingabe
Zug korrekt? Wenn ja, Ausführung
Spielende?
Feld bekannt?
Zug auswählen
Feld ist unbekannt

PROGRAMMLAUF

```
++Hexapawn (Sechsbauern-Spiel)++

Gespielt wird auf einem 3 x 3 - Brett,
welches zu Beginn so aussieht:

   O O O     O=Computer
   . . .
   X X X     X=Spieler

Jeder Spieler hat drei Bauern, welche
sich geradeaus auf ein leeres Feld
oder diagonal nach links bzw. rechts
auf ein vom Gegner besetztes Feld be-
wegen duerfen. Es gewinnt, wer alle
Bauern des Gegners schlaegt, wer den
Gegner zugunfaehig macht oder wer
zuerst einen Bauern auf die gegner-
ische Seite bringt.

Druecken Sie eine beliebige Taste!

Ihr Zug von 8
Ihr Zug nach 5

Mein Zug: 1-5

Ihr Zug von 9
Ihr Zug nach 5

Mein Zug: 3-6
```

Ihr Zug von 7
Ihr Zug nach 4

Mein Zug: 6-9

Ich habe gewonnen

Eine Taste druecken

VARIABLENLISTE

A(X)	:Spielfeld festhalten
F(X,Y)	:Tabelle der bekannten Spielfelder
V(X,Y), N(X,Y)	:Tabelle der Züge zu den bekannten Spielfeldern
F	:Anzahl der bekannten Spielfelder
ANF	:Anfangsadresse der Display-List
BILD	:Anfangsadresse des Bildschirmspeichers im Grafikmodus 1
G	:Indikator für am Zug befindlichen Spieler und somit auch Indikator für den Gewinner
V	:Feld, von dem der Spielerzug ausgeht
N	:Feld, auf das der Bauer des Spielers zieht
Z	:Anzahl der möglichen Züge des am Zug befindlichen Spielers
UNS,SYM	:Werden gebraucht, um herauszufinden, ob die Spielstellung mit einer abgespeicherten identisch bzw. symmetrisch ist.
S	:S=0 heißt, Spielstellung und abgespeicherte Stellung sind identisch S=1 heißt, Spielstellung und abgespeicherte Stellung sind zueinander symmetrisch.
R	:Nummer des ausgewählten Zuges des

	Computers.
V1,V2,N1,N2	:Dienen zum Ermitteln des symmetrischen Zuges
D1,D2	:Abspeichern des letzten Zuges des Computers

Listing: Hexapawn

```
1000 Rem +++ Initialisierung ++++++++++++++++++++++++++++++++++
1100 Rem --- Variablen und Felder
1110 Dim A(9),F(30,9),V(30,4),N(30,4)
1120 F=0
1200 Rem --- Spielfeld initialisieren
1210 For I=1 to 9
1220 If I<4 then A(I)=1
1230 If I>3 and I<7 then A(I)=0
1240 If I>6 then A(I)=-1
1250 Next I
2000 Rem +++ Hauptprogramm ++++++++++++++++++++++++++++++++++++
2100 Rem --- Ausgabe der Spielregeln
2110 Graphics 0: Setcolor 2,0,0: Setcolor 4,1,8
2120 Print "++ Hexapawn  (Sechsbauern-Spiel) ++"
2130 Print
2140 Print "Gespielt wird auf einem 3x3 Brett,"
2150 Print "welches zu Beginn so aussieht:"
2160 Print
2170 Print " O O O        O=Computer"
2180 Print " . . .                  "
2190 Print " X X X        X=Spieler "
2200 Print
2210 Print "Jeder Spieler hat drei Bauern, welche"
2220 Print "sich geradeaus auf ein leeres Feld"
2230 Print "oder diagonal nach links bzw. rechts"
2240 Print "auf ein vom Gegner besetztes Feld be-"
2250 Print "wegen duerfen. Es gewinnt, wer alle"
2260 Print "Bauern des Gegners schlaegt, wer den"
2270 Print "Gegner zugunfaehig macht oder wer"
2280 Print "zuerst einen Bauern auf die gegner-"
2290 Print "ische Seite bringt."
2300 Print
2310 Print "Druecken Sie eine beliebige Taste!"
2320 If Peek(753)=0 then 2320
```

```
2400 Rem --- Bildschirm setzen
2410 Graphics 1+16: Setcolor 4,0,0
2420 ANF=Peek(560)+256*Peek(561)
2430 Poke ANF+6,7
2440 Position 0,1: Print #6;"  ** HeXaPaWn **"
2450 BILD=Peek(88)+256*Peek(89)

2500 Rem --- Spielablauf
2510 Gosub 3000            :Rem Spielfeld zeichnen
2520 For I=8 to 0 Step -1: Sound 0,30+20*I,30,8
2530 For J=BILD+220 to BILD+239+20*I: Poke J,Peek(J+20): Next J
2540 Next I: Sound 0,0,0,0
2550 Gosub 4000            :Rem Spielerzug
2560 Gosub 3000            :Rem Spielfeld zeichnen
2570 Gosub 5000            :Rem Computerzug
2580 Goto 2510

3000 Rem +++ Spielfeld zeichnen ++++++++++++++++++++++++++++++++
3010 Position 0,3
3020 For I=1 to 3
3030 Print #6;"   ";3*I-2;"!";3*I-1;"!";3*I;"    ";
3040 For J=1 to 3
3050 If A(3*(I-1)+J)=-1 then print #6;"X";
3060 If A(3*(I-1)+J)=0 then print #6;" ";
3070 If A(3*(I-1)+J)=1 then print #6;"O";
3080 If J<3 then print #6;"!";
3090 Next J: Print #6
3100 If I<3 then print #6;"  -+-+-   -+-+-"
3110 Next I: Print #6
3120 Return

4000 Rem +++ Spielerzug ++++++++++++++++++++++++++++++++++++++++
4100 Rem --- Spielende?
4110 G=-1: Gosub 6000
4120 If A(7)=1 or A(8)=1 or A(9)=1 or Z=0 then 7000

4200 Rem --- Eingabe des Spielerzuges
4210 Position 0,11
4220 Print #6;"   ihr zug  von ";
4230 Poke 764,255: Open #2,4,0,"K:": Get #2,V: V=V-48: Close #2
4240 Print #6;V
```

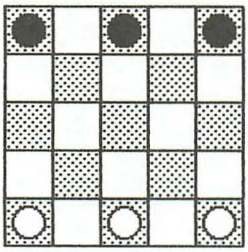
```
4250 Print #6;"   ihr zug nach ";
4260 Poke 764,255: Open #2,4,0,"K:": Get #2,N: N=N-48: Close #2
4270 Print #6;N
4280 If V<1 or V>9 or N<1 or N>9 then 4220

4300 Rem --- Ist Zug korrekt?
4310 For I=1 to Z
4320 If V=V(T,I) and N=N(T,I) then A(V)=0: A(N)=-1: Return
4330 Next I
4340 Goto 4220

5000 Rem +++ Computerzug ++++++++++++++++++++++++++++++++++++++++

5100 Rem --- Spielende?
5110 G=1: Gosub 6000
5120 If A(1)=-1 or A(2)=-1 or A(3)=-1 or Z=0 then 7000

5200 Rem --- Ist Spielfeld bekannt?
5210 If F=0 then 5500          :Rem Stellung ist unbekannt
5220 For T=1 to F
5230 UNS=0: SYM=0
5240 For I=0 to 6 Step 3
5250 For J=1 to 3
5260 If A(I+J)=F(T,I+J) then UNS=UNS+1
5270 If A(I+4-J)=F(T,I+J) then SYM=SYM+1
5280 Next J
5290 Next I
5300 If UNS=9 then S=0: Goto 5400
5310 If SYM=9 then S=1: Goto 5400
5320 Next T
5330 Goto 5500                 :Rem Stellung ist unbekannt

5400 Rem --- Spielfeld ist bekannt, Zug auswaehlen
5405 Position 3,18
5410 If V(T,1)+V(T,2)+V(T,3)+V(T,4)=0 then print #6;"ich gebe auf":
     Goto 7000
5420 R=Int(Rnd(0)*4)+1: If V(T,R)=0 then 5420
5430 V1=Int((V(T,R)-1)/3): V2=V(T,R)-3*V1
5440 N1=Int((N(T,R)-1)/3): N2=N(T,R)-3*N1
5450 If S=1 then V2=4-V2: N2=4-N2
5460 Print #6;" mein zug: ";3*V1+V2;"-";3*N1+N2
5470 A(3*V1+V2)=0: A(3*N1+N2)=1
5480 D1=T: D2=R                :Rem Letzter Zug speichern
5490 Return
```

```
5500 Rem --- Spielfeld ist unbekannt
5510 F=F+1: T=F: S=0
5520 For I=1 to 9
5530 F(F,I)=A(I)
5540 Next I
5550 Goto 5400

6000 Rem +++ Alle moeglichen Zuege ermitteln +++++++++++++++++++
6010 T=F+1: Z=0
6020 For I=1 to 4: V(T,I)=0: Next I
6030 For I=0 to 6 Step 3
6040 For J=1 to 3
6050 If I=3+3*G or A(I+J)<>G then 6110
6060 If A(I+J+3*G)=0 then Z=Z+1: V(T,Z)=I+J: N(T,Z)=I+J+3*G
6070 If J=1 then 6090
6080 If A(I+J-1+3*G)=-G then Z=Z+1: V(T,Z)=I+J: N(T,Z)=I+J-1+3*G
6090 If J=3 then 6110
6100 If A(I+J+1+3*G)=-G then Z=Z+1: V(T,Z)=I+J: N(T,Z)=I+J+1+3*G
6110 Next J
6120 Next I
6130 Return

7000 Rem +++ Spielende ++++++++++++++++++++++++++++++++++++++++
7010 For I=15 to 0 Step -.2: Sound 0,10+10*Rnd(0),0,I: Sound 1,
     100+10*Rnd(0),16,I: Setcolor 4,0,16*Rnd(0): Next I
7015 Setcolor 4,0,0
7020 For I=11 to 19: Position 0,I: Print #6;"
                                                  ": Next I
7030 Position 1,11
7040 If G=-1 then print #6;"ich habe gewonnen"
7050 If G=1 then print #6;"sie haben gewonnen": V(D1,D2)=0
7060 Position 0,14
7070 Print #6;"eine Taste druecken"
7080 A=Peek(BILD+20)
7090 For I=BILD+20 to BILD+38
7100 Poke I,Peek(I+1): If Peek(753)=3 then 1200
7110 Next I: Poke BILD+39,A
7120 Goto 7080
```

```
  -!-!-!-!-
3  ! ! ! !
  -!-!-!-!-
2  ! ! ! !
  -!-!-!-!-
1 X! !X! !X
  1 2 3 4 5

Mein Zug: Von 5,5 nach 4,4


5 O! !O! !
  -!-!-!-!-
4  ! ! !O!
  -!-!-!-!-
3  ! ! ! !
  -!-!-!-!-
2  ! ! ! !
  -!-!-!-!-
1 X! !X! !X
  1 2 3 4 5

Ihr Zug: Von   ?3,1
```

SPIELREGELN

Mini-Dame ist eine vereinfachte Version des bekannten Dame-Spieles. Zwei Spieler, Weiß und Schwarz, ziehen abwechselnd. Gespielt wird auf einem 5x5 - Brett mit je drei Steinen. Es wird nur auf den schwarzen Feldern gespielt:



Bei einem Zug kann schräg um ein Feld weiter auf ein leeres Feld gezogen werden, Rückwärtsschieben ist nicht möglich. Befindet sich ein gegenerischer Stein unmittelbar vor dem eigenen Feld und ist das Feld hinter diesem Stein leer, so darf der gegnerische Stein übersprungen werden. Dieser Stein wird dann vom Spielbrett entfernt.

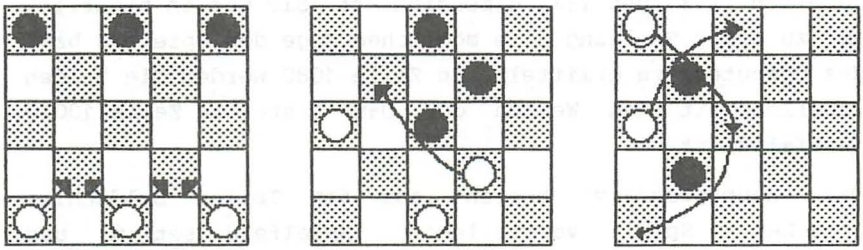
Es herrscht Sprungzwang, d.h. kann gesprungen werden, so muß auch gesprungen werden. Bestehen mehrere Möglichkeiten zum Springen, so kann eine davon ausgesucht werden. Mehrmaliges Springen ist möglich.

Gelangt ein Stein in die gegenüberliegende Reihe, so wird er zu einer Dame. Eine Dame besteht aus zwei

übereinanderliegenden Steinen und hat gegenüber den sonstigen Spielsteinen einige Vorteile. Zum einen darf sich die Dame auch rückwärts bewegen. Außerdem kann sie beliebig viele leere Felder vorwärtsrutschen, um einen gegnerischen Stein zu überspringen; vorausgesetzt natürlich daß das Feld unmittelbar dahinter frei ist.

Das Spiel endet, wenn alle Steine eines Spielers geschlagen sind oder wenn ein Spieler nicht mehr ziehen kann.

Hier einige Beispiele für mögliche Züge, Weiß sei immer am Zuge:



PROGRAMMERLÄUTERUNG

"Mini-Dame" übertrifft an Komplexität und Interessantheit die beiden vorangegangenen Lernspiele "Bauernkampf" und "Hexapawn" bei weitem. Dementsprechend ist auch das Programm äußerst umfangreich und es bedarf einer genauen Beschreibung, um die Funktionsweise des Programmes richtig verstehen zu können.

Die "Initialisierung" befindet sich in den Zeilen 1000 bis

1080. Das Spielfeld wird in der Tabelle A(X,Y) abgespeichert. A(1,1) ist das Feld links unten, A(1,5) das Feld links oben und entsprechend A(5,1) das Feld rechts unten. Die Tabellen F(X,Y), V(X,Y) und N(X,Y) werden wie schon gewohnt zur Abspeicherung der verschiedenen Spielstellungen und der zugehörigen Züge verwendet. Vorgesehen sind 50 Stellungen mit maximal 8 Zügen zu jeder Stellung. Die Variable F gibt die Anzahl der abgespeicherten und damit bekannten Spielfelder an, die Variable G wird als Indikator für den am Zug befindlichen Spieler verwendet. G=1 bedeutet Spieler am Zuge, G=1 heißt Computer am Zuge. Die Tabellen B(X,Y), C(X,Y) dienen zum Zwischenspeichern des Spielfeldes. Ihre Funktion wird später deutlich. Ferner werden in Zeile 1070 die beiden Tabellen X(X) und Y(X) dimensioniert. Sie werden benötigt, um zu einer Stellung alle möglichen Züge des Spielers bzw. des Computers zu ermitteln. In Zeile 1080 werden die beiden Tabellen mit den Werten der Data-Liste in Zeile 10010 initialisiert.

Das "Hauptprogramm" besteht aus den Teilen "Bildschirm loeschen, Spiel vorstellen", "Spielfeld setzen" und "Spielablauf". Der "Spielablauf" ist so ausgelegt, daß der Computer immer beginnt. In ihm werden die Unterprogramme "Spielfeld zeichnen", "Computerzug" und "Spielerzug" aufgerufen.

Das Spielfeld wird in den Zeilen 3000 bis 3150 ausgegeben. Wichtig ist hierbei allein, wie die verschiedenen Spielfiguren gespeichert werden. Die Zeilen 3050 bis 3090 geben hierüber unmittelbar Auskunft. Zunächst sei erwähnt, daß die Steine des Spielers durch ein "X" repräsentiert werden; die Steine des Computers werden durch ein "O" repräsentiert. Zur Unterscheidung von einfachen Steinen und Damensteinen werden Groß- und Kleinbuchstaben verwendet. Großbuchstaben bedeuten einfache Spielsteine, Kleinbuchstaben bedeuten Damesteine. Eine -2 steht nun für einen Damestein des Spielers, eine -1 für einen einfachen Stein des Spielers.

Leere Felder werden durch eine Null markiert. Eine 1 bzw. eine 2 steht für einen einfachen Stein bzw. einen Damestein des Computers. Diese Kodierung des Spielfeldes ist natürlich völlig frei dem Programmierer überlassen. Es bieten sich manchmal jedoch bestimmte Werte an, gerade wenn manche Programmteile gleichermaßen für Spieler und Computer verwendet werden.

Der Einfachheit halber wird zunächst der "Spielerzug" erklärt. Zu Beginn wird analog zum vorangegangenen Programm "Hexapawn" getestet, ob das Spiel zu Ende ist. Dazu wird das Unterprogramm "Berechnung aller möglichen Züge Spieler G" in Zeile 6000 aufgerufen. Dieses Unterprogramm ermittelt alle zulässigen Züge und speichert diese ab. Zunächst wird die Variable Z auf Null gesetzt, danach wird das gesamte Spielfeld abgetastet. Dazu dienen die zwei Schleifen in den Zeilen 6040/6050. Wird ein einfacher Stein von Spieler G gefunden, so ermitteln die Zeilen 6200 bis 6290 alle möglichen Züge dieses Steines. Dasselbe machen die Zeilen 6300 bis 6430 für einen Damestein von Spieler G. Die Variable Z gibt die Anzahl der möglichen Züge an. Jetzt jedoch muß folgendes beachtet werden: Besteht eine Möglichkeit zum Springen, so muß auch gesprungen werden. In den Zeilen 6110/6120 wird geprüft, ob ein überspringen möglich ist. Wenn ja, müssen alle abgespeicherten Züge, die kein Springen darstellen, gelöscht werden. Dies geschieht in den Zeilen 6130 und 6140. Anschließend werden die Züge geordnet, so daß die möglichen Züge am Anfang der Tabelle stehen, d.h. von V(T,1) bis V(T,Z). Ein Spielende liegt genau dann vor, wenn in Zeile 4120 Z=0 ist. In diesem Falle springt der Computer nach Zeile 9000.

Ist Z größer Null, so kann der Spieler ziehen, das Spiel geht also weiter. Nun erfolgt die Eingabe des Zuges vom Spieler. Dazu dienen die Zeilen 4200 bis 4260.

Nun muß natürlich getestet werden, ob der Zug überhaupt zulässig ist. Dazu wird der vom Spieler eingegebene Zug einfach mit den abgespeicherten zulässigen Zügen

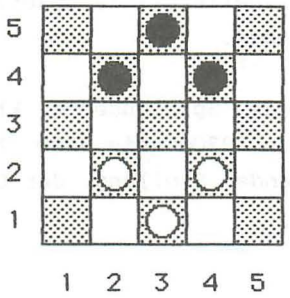
verglichen. Stimmt der Zug mit einem von diesen abgespeicherten Zügen überein, so ist der Zug korrekt und es wird nach Zeile 4400 gesprungen. Ich sehe gerade, daß ich noch nachzutragen habe, wie die Züge abgespeichert werden.
Ein Feld wird ja durch zwei Koordinaten festgelegt, einer X -und einer Y-Koordinate. Für einen Zug sind deshalb vier Koordinaten vonnöten, zwei für das Ausgangsfeld und zwei für das Zielfeld. Für die Abspeicherung der Züge müßten deshalb vier Tabellen verwendet werden. Um dies zu vermeiden, wird das Spielfeld auf folgende Weise numeriert:

21	22	23	24	25
16	17	18	19	20
11	12	13	14	15
6	7	8	9	10
1	2	3	4	5

Dem Feld X,Y entspricht damit die Zahl X+(Y-1)*5. Auf diese Weise werden nur zwei statt vier Tabellen benötigt.

Ist der Zug korrekt, so wird er anschließend ausgeführt. Dazu dient das Unterprogramm in den Zeilen 7000 bis 7050. Nun erfolgt eine Prüfung, ob nochmaliges Springen möglich ist. Dazu werden noch einmal alle möglichen Züge ermittelt. Ist Z=0 oder aber ist Abs(NX-VX)^^2, d.h. ist überhaupt kein Zug mehr möglich oder aber ist der Spieler gar nicht gesprungen, so erfolgt in Zeile 4510 ein sofortiger Rücksprung; der Zug des Spielers ist dann beendet.
Ansonsten muß getestet werden, ob mit dem zuletzt gesprungenen Stein noch ein Sprung möglich ist. Wenn ja, muß der Spieler zusätzlich zwei Koordinaten angeben und der Zug wird in den Zeilen 4600 bis 4650 auf seine Korrektheit überprüft und ausgeführt.

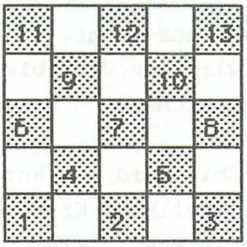
Der "Computerzug" befindet sich in den Zeilen 5000 bis 5880. Wie beim "Spielerzug" auch, wird zunächst getestet, ob das Spiel zu Ende ist. Wenn nein, muß geprüft werden, ob die vorliegende Spielstellung schon bekannt ist. Das kennen wir ja bereits schon aus den vorangegangenen Programmen. Bei der Komplexität von "Mini-Dame" ist ein Erkennen von symmetrischen Stellungen natürlich unerläßlich. Hierzu dienen wieder die beiden Variablen UNS und SYM.
Die Tabelle F(X,Y) wird Ihnen vieleicht schon aufgefallen sein. Um Speicherplatz zu sparen, werden nur die schwarzen Felder abgespeichert, da ja nur auf diesen Feldern gespielt wird. Dadurch benötigt eine Spielstellung 13 Werte. Das Spielfeld wird also so abgespeichert:



Ist die Spielstellung unbekannt, so springt der Computer nach Zeile 5800. Hier wird die Variable F incrementiert und das Spielfeld abgespeichert. Das ist ja bei allen Lernprogrammen gleich.
In Zeile 5400 ist das Spielfeld nun auf jeden Fall bekannt, d.h. es stimmt mit Stellung Nr. Q überein. Die Auswahl eines Zuges erfolgte bisher immer völlig zufällig. Bei einem Spiel wie "Mini-Dame" ist dies äußerst uneffektiv. Damit der Computer lernen würde gut zu spielen, müßte er

sehr viele Spiele bestreiten. Besser ist es, den verschiedenen Zügen des Computers Bewertungen zu geben und anschließend den Zug mit der höchsten Bewertung auszuführen. Dies wurde in diesem Programm verwirklicht. Die Variable MAX gibt die maximale Bewertung an. Die Tabelle B(X,Y) dient zum Zwischenspeichern des Spielfeldes, das wurde ja bereits erwähnt. Wurde der Zug des Computers ausgeführt, so wird die entstandene Stellung in den Zeilen 8000 bis 8130 bewertet. Hier wird die Anzahl der Spielsteine des Computers und des Spielers berücksichtigt, außerdem hat die Anzahl der möglichen Züge derselben einen Einfluß. Ebenso wird geprüft, ob der Spieler springen kann. Ist die Bewertung BEW größer als MAX, so wird dieser Zug abgespeichert. Dazu dienen die Variablen G1-G6. Außerdem wird das zugehörige Spielfeld in der Tabelle C(X,Y) gespeichert. Nachdem auf diese Weise alle Züge des Computers ausgeführt wurden, wird das Spielfeld anschließend mit der Tabelle C(X,Y) gesetzt und der Zug des Computers wird ausgegeben.

Nun bleibt nur noch das "Spielende" zu klären. Interessant sind hier die Zeilen 9200 bis 9280. Zu ihrer Erklärung betrachten Sie folgende Stellung, der Computer (Schwarz) sei am Zuge:



Nehmen wir an, der Computer führt den Zug 2,4 nach 1,3 aus. Bei richtigem Weiterspielen des Spielers führt dies zum Verlust des Spieles. Der Computer entfernt also nach mehrmaligem Verlieren diesen Zug. Da der Zug 4,4 nach 5,3 ja eigentlich genau der gleiche Zug ist (nur seitenvertauscht), könnte auch gleich dieser Zug entfernt werden. Dazu dienen die Zeilen 9200 bis 9280. Ist die Spielstellung vor dem Zug (sie ist in B(X,Y) gespeichert) symmetrisch, so wird auch gleich der symmetrische Zug entfernt.

Aufgabe: Erweitern Sie das Programm so, daß sowohl der Computer als auch der Spieler den ersten Zug machen kann.

Die Wahl bleibt natürlich dem Spieler überlassen.

WEITERFÜHRENDE TIPS UND TRICKS

Ich hoffe, daß das Programm "Mini-Dame" Ihnen gezeigt hat, daß die Lernmethode nicht nur auf triviale Spiele sondern auch auf interessante komplexe Spiele angewendet werden kann. Die Kombination von heuristischen Verfahren und der Lernmethode erweist sich dabei als äußerst wirkungsvoll. Das Ausschlaggebende dabei ist, daß der Computer lernt, optimal zu spielen

Der erste Tip, den ich geben möchte, betrifft die Abspeicherung der Spielstellungen und der zugehörigen Züge. In diesem Buch erfolgte diese Abspeicherung immer mittels vorher dimensionierter Tabellen. Das war notwendig, damit die Programme gut verstanden werden können und nicht zu unübersichtlich werden. Diese Methode der Dimensionierung ist jedoch im Hinblick auf den nötigen Speicherplatz äußerst verschwenderisch. Besser ist es, die Stellungen und

die Züge in einem freien Speicherbereich mittels der Basic-Befehle Peek und Poke zu speichern. Das spart außerordentlich viel Speicherplatz, ein Beispiel mag dies verdeutlichen:
Es sollen 100 Spielfelder mit je 10 Werten abgespeichert werden. Die Anweisung Dim F(100,10) benötigt ca. 7 KByte Speicherplatz, während man mit Peek und Poke mit 1 KByte auskommt.

Falls das Spiel verloren wurde, wurde bisher immer nur der letzte Zug des Computers entfernt. Wenn nach dem Entfernen dieses Zuges zu dieser Stellung kein Zug mehr möglich ist (alle Perlen entfernt), so kann der Zug, der zu dieser Stellung geführt hat (also der vorletzte Zug) ebenfalls entfernt werden und das Ganze wiederholt sich. Um dies auszunützen müssen sämtliche Züge des Computers abgespeichert werden.

Traf der Computer auf eine Stellung, bei der "alle Perlen entfernt" waren, so gab der Computer bisher immer auf. Man kann den Computer jedoch ebenso weiterspielen lassen. Macht man das, so bietet sich folgende einfache Möglichkeit des Kommentars der Spielerzüge an: Führt ein Zug des Spielers zu einer Stellung, bei der kein Zug mehr möglich ist, so war dieser Zug ganz offensichtlich gut. Der Computer könnte dann ausgeben "Dieser Zug war gut". Ein Kommentar "Dieser Zug war schlecht" kann ebenfalls ausgegeben werden. Nämlich dann, wenn ein Zug des Spielers von einer Stellung "ohne Perlen" zu einer Stellung "mit Perlen" führt.

PROGRAMMLAUF

```
+++ Mini-Dame +++


5 O! !O! !O                    5  ! ! ! !O
-!-!-!-!-                      -!-!-!-!-
4  ! ! ! !                     4  !O! !O!
-!-!-!-!-                      -!-!-!-!-
3  ! ! ! !                     3  ! ! ! !
-!-!-!-!-                      -!-!-!-!-
2  ! ! ! !                     2  !X! ! !
-!-!-!-!-                      -!-!-!-!-
1 X! !X! !X                    1  ! !X! !X
  1 2 3 4 5                      1 2 3 4 5

Mein Zug: Von 1,5 Nach 2,4     Ihr Zug: Von ? 2,2
                                        Nach? 3,3


5  ! !O! !O                    5  ! ! ! !O
-!-!-!-!-                      -!-!-!-!-
4  !O! ! !                     4  !O! !O!
-!-!-!-!-                      -!-!-!-!-
3  ! ! ! !                     3  ! !X! !
-!-!-!-!-                      -!-!-!-!-
2  ! ! ! !                     2  ! ! ! !
-!-!-!-!-                      -!-!-!-!-
1 X! !X! !X                    1  ! !X! !X
  1 2 3 4 5                      1 2 3 4 5

Ihr Zug: Von ? 1,1             Mein Zug: Von 4,4 Nach 2,2
Nach? 2,2
```

```
5  ! !O! !O
-!-!-!-!-
4  !O! ! !
-!-!-!-!-
3  ! ! ! !
-!-!-!-!-
2  !X! ! !
-!-!-!-!-
1  ! !X! !X
  1 2 3 4 5
```

Mein Zug: Von 3,5 Nach 4,4

```
5  ! ! ! !O
-!-!-!-!-
4  !O! ! !
-!-!-!-!-
3  ! ! ! !
-!-!-!-!-
2  !O! ! !
-!-!-!-!-
1  ! !X! !X
  1 2 3 4 5
```

Ihr Zug: Von ? 3,1
Nach? 1,3
+Nach? 3,5

usw.

VARIABLENLISTE

A(X,Y)	:Spielfeld
F(X,Y)	:Speicherung der bekannten Spielfelder
V(X,Y), N(X,Y)	:Tabellen der Züge
B(X,Y), C(X,Y)	:Zwischenspeichern des Spielfeldes
X(X), Y(X)	:Tabellen zur Ermittlung aller möglichen Züge
F	:Anzahl bekannter Spielfelder
G	:Indikator für am Zug befindlichen Spieler
Z	:Anzahl der möglichen Züge
VX,VY,NX,NY	:Zug des Spielers
UNS,SYM	:Zeiger für Identität/Symmetrie
MAX	:Ermitteln der höchsten Bewertung der Computerzüge
BEW	:Bewertung einer Stellung
D1, D2	:Speichern des letzten Computerzuges
G1,G2,G3,G4,G5,G6	:Zug des Computers

Listing: Mini-Dame

```
1000 Rem +++ Initialisierung ++++++++++++++++++++++++++++++++++++++
1010 Dim A(5,5)                :Rem Spielfeld
1020 Dim F(50,13)              :Rem Bekannte Felder
1030 Dim V(50,8), N(50,8)      :Rem Tabellen fuer Zuege
1040 F=0                       :Rem Anzahl bekannter Felder
1050 G=0                       :Rem Ind. f. am Zug befindl. Spieler
1060 Dim B(5,5), C(5,5)        :Rem Zwischenspeichern des Spielfeldes
1070 Dim X(4), Y(4)            :Rem Tab. z. Ermittl. aller moegl. Zue.
1080 For I=1 to 4: Read X,Y: X(I)=X: Y(I)=Y: Next I

2000 Rem +++ Hauptprogramm ++++++++++++++++++++++++++++++++++++++
2100 Rem --- Bildschirm loeschen, Spiel vorstellen
2110 Graphics 0
2120 Print "+++ Mini-Dame +++"
2130 Print
2200 Rem --- Spielfeld setzen
2210 For I=1 to 5
2220 For J=1 to 5
2230 A(J,I)=0
2240 Next J
2250 Next I
2260 A(1,1)=-1: A(3,1)=-1: A(5,1)=-1
2270 A(1,5)=1: A(3,5)=1: A(5,5)=1
2300 Rem --- Spielablauf
2310 Gosub 3000                :Rem Spielfeld zeichnen
2320 Gosub 5000                :Rem Computerzug
2330 Gosub 3000                :Rem Spielfeld zeichnen
2340 Gosub 4000                :Rem Spielerzug
2350 Goto 2310

3000 Rem +++ Spielfeld ausgeben ++++++++++++++++++++++++++++++++++
3010 Print
3020 For I=5 to 1 step -1
3030 Print I;" ";
3040 For J=1 to 5
```

```
3050 If A(J,I)=-2 then print "x";
3060 If A(J,I)=-1 then print "X";
3070 If A(J,I)=0  then print " ";
3080 If A(J,I)=1  then print "O";
3090 If A(J,I)=2  then print "o";
3100 If J<5 then print "!";
3110 Next J: Print
3120 If I>1 then print "  -!-!-!-!-"
3130 Next I: Print
3140 Print "  1 2 3 4 5": Print
3150 Return

4000 Rem +++ Spielerzug ++++++++++++++++++++++++++++++++++++++++

4100 Rem --- Spielende?
4110 G=-1: Gosub 6000
4120 If Z=0 then 9000

4200 Rem --- Zugeingabe
4210 Print "Ihr Zug: Von  ";
4220 Input VX, VY
4230 If VX<1 or VY<1 or VX>5 or VY>5 then 4210
4240 Print "          Nach ";
4250 Input NX, NY
4260 If NX<1 or NY<1 or NX>5 or NY>5 then 4240

4300 Rem --- Ist Zug korrekt?
4310 For I=1 to Z
4320 If Abs(N(T,I)-V(T,I))>6 and Abs(NX-VX)=1 then print "Sie
     muessen springen!": Goto 4200
4330 If VX+(VY-1)*5=V(T,I) and NX+(NY-1)*5=N(T,I) then 4400
4340 Next I
4350 Print "Ihr Zug ist unkorrekt!": Goto 4200

4400 Rem --- Zug ausfuehren
4410 Gosub 7000

4500 Rem --- Noch einmal springen?
4510 Gosub 6000: If Z=0 or Abs(NX-VX)<>2 then Return
4520 For I=1 to Z
4530 If Abs(N(T,I)-V(T,I))<8 or V(T,I)<>NX+(NY-1)*5 then next I:
     Return
4540 VX=NX: VY=NY
4550 Print "          +Nach ";
```

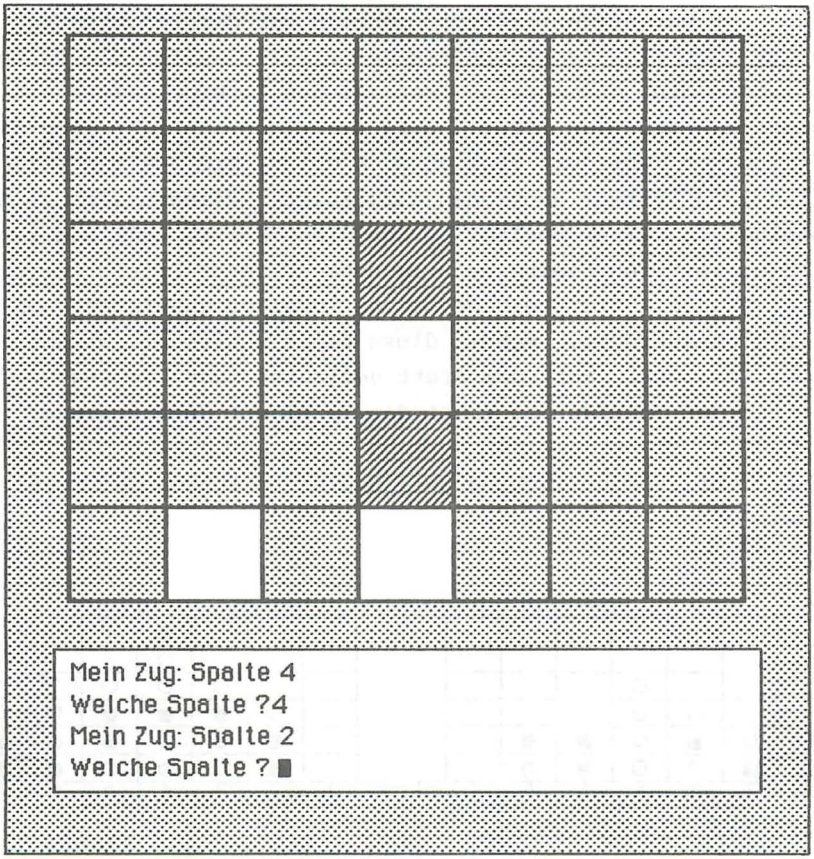
```
4560 Input NX, NY
4570 If NX<1 or NY<1 or NX>5 or NY>5 then 4550
4600 Rem --- Zusaetzlicher Zug ausfuehren
4610 If Abs(NX-VX)<>2 then print "Sie muessen noch einmal springen":
     Goto 4550
4620 For I=1 to 8
4630 If V(T,I)=VX+(VY-1)*5 and N(T,I)=NX+(NY-1)*5 then gosub 7000:
     Return
4640 Next I
4650 Print "Zug unkorrekt!": Goto 4550

5000 Rem +++ Computerzug ++++++++++++++++++++++++++++++++++++++++
5100 Rem --- Spielende?
5110 G=1: Gosub 6000
5120 If Z=0 then 9000
5200 Rem --- Spielfeld bekannt?
5210 If F=0 then 5800      :Rem Unbekannt
5220 For Q=1 to F
5230 UNS=0: SYM=0: P=0
5240 For I=1 to 5
5250 For J=1 to 5
5260 If (I+J)/2<>Int((I+J)/2) then 5300
5270 P=P+1: If A(J,I)<>F(Q,P) and A(6-J,I)<>F(Q,P) then J=5: I=5:
     Goto 5300
5280 If A'J,I)=F(Q,P) then UNS=UNS+1
5290 If A(6-J,I)=F(Q,P) then SYM=SYM+1
5300 Next J
5310 Next I
5320 If UNS=13 or SYM=13 then 5400
5330 Next Q
5340 Goto 5800             :Rem Unbekannt
5400 Rem --- Zug auswaehlen
5410 P=0: For I=1 to 8: P=P+V(Q,I): Next I: If P=0 then print
     "Ich gebe auf!!": Goto 9000
5420 For I=1 to 5: For J=1 to 5: B(J,I)=A(J,I): Next J: Next I
5430 MAX=-500
5440 For A=1 to 8
5450 If V(Q,A)=0 then 5600
```

```
5460 For I=1 to 5: For J=1 to 5: A(J,I)=B(J,I): Next J: Next I
5470 VY=Int((V(Q,A)+4)/5): VX=V(Q,A)-5*(VY-1)
5480 NY=Int((N(Q,A)+4)/5): NX=N(Q,A)-5*(NY-1)
5490 If SYM=13 then VX=6-VX: NX=6-NX
5495 X1=VX: X2=NX: Y1=VY: Y2=NY
5500 Gosub 7000: Gosub 6000
5510 If Z=0 or Abs(NX-VX)<>2 then X3=0: Y3=0: Goto 5570
5520 For I=1 to 8
5530 If V(T,I)<>NX+(NY-1)*5 or Abs(V(T,I)-N(T,I))<8 then next I:
     X3=0: Y3=0: Goto 5570
5540 VX=NX: VY=NY
5550 NY=Int((N(T,I)+4)/5): NX=N(T,I)-5*(NY-1): X3=NX: Y3=NY
5560 Gosub 7000
5570 Gosub 8000                    :Rem Stellung bewerten
5580 If (BEW=MAX and Rnd(0)>.4) or BEW<MAX then 5600
5590 MAX=BEW: D2=A: G1=X1: G2=Y1: G3=X2: G4=Y2: G5=X3: G6=Y3
5595 For I=1 to 5: For J=1 to 5: C(J,I)=A(J,I): Next J: Next I
5600 Next A
5610 For I=1 to 5: For J=1 to 5: A(J,I)=C(J,I): Next J: Next I

5700 Rem --- Zug ausgeben
5710 Print "Mein Zug: Von ";G1;",";G2;" Nach ";G3;",";G4
5720 If G5>0 then print "                    +Nach ";G5;",";G6
5730 D1=Q: Return

5800 Rem --- Spielfeld ist unbekannt
5810 F=F+1: Q=F: UNS=0: P=0
5820 For I=1 to 5
5830 For J=1 to 5
5840 If (I+J)/2<>Int((I+J)/2) then 5860
5850 P=P+1: F(Q,P)=A(J,I)
5860 Next J
5870 Next I
5880 Goto 5400

6000 Rem +++ Berechnung aller moeglichen Zuege Spieler G ++++++++++

6010 Rem --- Schleifendurchlauf
6020 T=F+1: Z=0
6030 For I=1 to 8: V(T,I)=0: Next I
6040 For I=1 to 5
6050 For J=1 to 5
6060 If A(J,I)=G then gosub 6200
```

```
6070 If A(J,I)=2*G then gosub 6300
6080 Next J
6090 Next I
6100 If Z=0 then Return
6110 P=0: For I=1 to Z: If Abs(N(T,I)-V(T,I))>6 then P=1
6120 Next I: If P=0 then Return
6130 For I=1 to Z: If Abs(N(T,I)-V(T,I))<8 then V(T,I)=0
6140 Next I
6150 Z=0: For I=1 to 8: If V(T,I)>0 and Z+1=I then Z=Z+1: Goto 6170
6160 If V(T,I)>0 then Z=Z+1: V(T,Z)=V(T,I): N(T,Z)=N(T,I): V(T,I)=0
6170 Next I
6180 Return

6200 Rem --- Einfacher Zug
6210 If J=1 then 6250
6220 If A(J-1,I-G)=0 then Z=Z+1: V(T,Z)=J+(I-1)*5: N(T,Z)=J-1+
     (I-G-1)*5: Goto 6250
6230 If J=2 or I=3-G then 6250
6240 If (A(J-1,I-G)=-G or A(J-1,I-G)=-2*G) and A(J-2,I-G-G)=0 then
     Z=Z+1: V(T,Z)=J+(I-1)*5: N(T,Z)=J-2+(I-G-G-1)*5
6250 If J=5 then Return
6260 If A(J+1,I-G)=0 then Z=Z+1: V(T,Z)=J+(I-1)*5: N(T,Z)=J+1+
     (I-G-1)*5: Return
6270 If J=4 or I=3-G then Return
6280 If (A(J+1,I-G)=-G or A(J+1,I-G)=-2*G) and A(J+2,I-G-G)=0 then
     Z=Z+1: V(T,Z)=J+(I-1)*5: N(T,Z)=J+2+(I-G-G-1)*5
6290 Return

6300 Rem --- Damezug
6310 For L=1 to 4
6320 X=X(L): Y=Y(L)
6330 For M=1 to 3
6340 If J+X*M<1 or J+X*M>5 or I+Y*M<1 or I+Y*M>5 then 6370
6350 If A(J+X*M,I+Y*M)=0 then next M: Goto 6380
6360 If A(J+X*M,I+Y*M)<>G and A(J+X*M,I+Y*M)<>2*G then 6390
6370 If M=1 then 6420
6380 Z=Z+1: V(T,Z)=J+(I-1)*5: N(T,Z)=J+X+(I+Y-1)*5: Goto 6420
6390 If J+X*M=1 or J+X*M=5 or I+Y*M=1 or I+Y*M=5 then 6370
6400 If A(J+X*M+X,I+Y*M+Y)<>0 then 6370
6410 Z=Z+1: V(T,Z)=J+(I-1)*5: N(T,Z)=J+X*M+X+(I+Y*M+Y-1)*5
6420 Next L
6430 Return
```

```
7000 Rem +++ Zug VX,VY nach NX,NY ausfuehren +++++++++++++++++++++
7010 A(NX,NY)=A(VX,VY): A(VX,VY)=0
7020 If NY=3-2*G then A(NX,NY)=2*G
7030 X=NX-1: If NX<VX then X=NX+1
7040 Y=NY-1: If NY<VY then Y=NY+1
7050 A(X,Y)=0: Return

8000 Rem +++ Stellung bewerten ++++++++++++++++++++++++++++++++++++
8010 BEW=0
8020 For I=1 to 5: For J=1 to 5
8030 If A(J,I)=G then BEW=BEW+50
8040 If A(J,I)=2*G then BEW=BEW+80
8050 If A(J,I)=-G then BEW=BEW-50
8060 If A(J,I)=-2*G then BEW=BEW-80
8070 Next J: Next I
8080 G=1: Gosub 6000: BEW=BEW+Z*2
8090 G=-1: Gosub 6000: BEW=BEW-Z: G=1
8100 For I=1 to Z
8110 If Abs(N(T,I)-V(T,I))>6 then BEW=BEW-10
8120 Next I
8130 Return

9000 Rem +++ Spielende ++++++++++++++++++++++++++++++++++++++++++++
9100 Rem --- Gewinner ausgeben
9110 If G=-1 then print "Ich habe gewonnen!": Goto 9300
9120 Print "Sie haben gewonnen!"
9130 V(D1,D2)=0
9200 Rem --- Auch symmetrischen Zug entfernen?
9210 For I=1 to 5: For J=1 to 5
9220 If B(J,I)<>B(6-J,I) then 9300
9230 Next J: Next I
9240 For I=1 to 8
9250 VY=Int((V(D1,I)+4)/5): VX=V(D1,I)-(VY-1)*5
9260 NY=Int((N(D1,I)+4)/5): NX=N(D1,I)-(NY-1)*5
9270 If G1=6-VX and G2=VY and G3=6-NX and G4=NY then V(D1,I)=0:
     Goto 9300
9280 Next I
9300 Rem --- Warteschleife, Neustart
9310 For I=1 to 500: Next I: Goto 2000
```

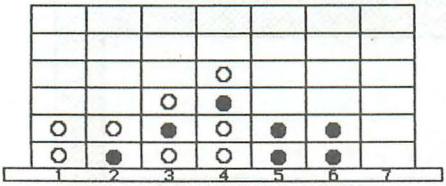
```
10000 Rem +++ Datas fuer Damezug ++++++++++++++++++++++++++++++++
10010 Data -1,1,1,1,1,-1,-1,-1
```

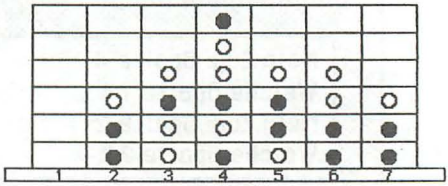

Mein Zug: Spalte 4
Welche Spalte ?4
Mein Zug: Spalte 2
Welche Spalte ?

SPIELREGELN

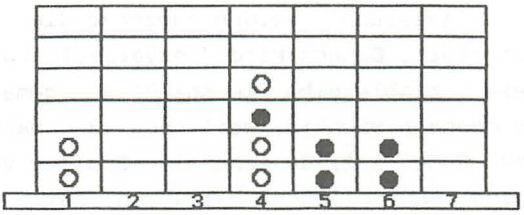
"Vier Gewinnt" ist ein weit verbreitetes und sehr beliebtes Spiel, welches im Spielwarenhandel erhältlich ist. Gespielt wird auf einem 7x6 - Brett, welches jedoch nicht wie üblich auf der Unterlage aufliegt, sondern auf der längeren Kante steht. In die sieben Spalten des Brettes werfen die beiden Spieler nun abwechselnd verschieden farbige Spielsteine. Dadurch, daß das Brett auf der Kante steht, fallen die Steine so weit wie möglich nach unten. Die Spalten werden also von unten nach oben aufgefüllt. Befinden sich sechs Steine in einer Spalte, so ist diese Spalte voll und kann keine weiteren Steine mehr aufnehmen. Jeder Spieler strebt nun danach, vier eigene Steine in einer ununterbrochenen Linie anzuordnen, wobei diese Linie senkrecht, waagrecht oder diagonal auf dem Brett verlaufen kann. Gelingt dies keinem der Spieler, so endet das Spiel mit Unentschieden.

Hier zur Verdeutlichung zwei Gewinnstellungen für Spieler "Weiß":





Betrachten Sie zur Einführung folgende Stellung, Weiß sei am Zuge:



Haben Sie etwas bemerkt? Weiß kann gewinnen, wenn Weiß auf folgende Weise vorgeht:

-Weiß füllt Spalte 3 auf
-Schwarz hat keine andere Wahl und muß Spalte 2 auffüllen
-Weiß wählt nun Spalte 2
-Schwarz muß wieder Spalte 3 auffüllen
-Weiß gewinnt durch Auffüllen von Spalte 3

Die entstandene Stellung sehen Sie oben links abgebildet!

PROGRAMMERLÄUTERUNG

Dieses Programm stellt einen Höhepunkt des Buches dar. Der Computer verwendet das "Alpha-Beta-Pruning", wobei die Suchtiefe vom Spieler wählbar ist! Das Programm ist dementsprechend äußerst komplex und bedarf zum richtigen Verständnis einer guten Beschreibung. Sie werden feststellen, daß der Computer schon in Suchtiefe 1 außergewöhnlich gut spielt. Jedoch liegt die Denkzeit des Computers schon in Suchtiefe 1 bei 3 bis 4 Minuten. Das ist natürlich zu langsam. Das Programm wird deshalb anschließend erweitert und mit einem Maschinensprache -Unterprogramm ausgestattet, welches die Denkzeit des Computers in Suchtiefe 1 auf ca. 30 Sekunden senkt. Zu beachten ist dabei, daß diese Zeit während des Spieles aus später erkennbaren Gründen wieder anwächst.

In der "Initialisierung" werden zunächst die notwendigen Felder dimensioniert. Danach wird die Variable ZUG auf Null gesetzt. Diese Variable gibt die Anzahl der gemachten Züge an. Das ist deshalb wichtig, weil nach 42 gemachten Zügen das Spiel zu Ende ist, da dann alle Spalten vollständig aufgefüllt sind. Anschließend wird in Zeile 1210 das Spielfeld gelöscht. Dazu werden alle Tabellenelemente auf Null gesetzt. Eine Null bezeichnet somit also ein leeres Feld, eine Eins bezeichnet einen Spielstein des Spielers und eine Fünf einen Spielstein des Computers. Die Tabelle B(X) wird auf 1 gesetzt. Diese Tabelle gibt zu jeder Spalte das unterste leere Feld an, wobei das Argument die Spalte bezeichnet. Ist ein Wert dieser Spalte gleich 7, so bedeutet dies, daß diese Spalte voll ist.

Das "Hauptprogramm" umfaßt die Teile "Spielfeld zeichnen", "Eingabe wer anfaengt", "Eingabe Schwierigkeitsgrad", "Spielablauf" und "Spielende". Zu erwähnen ist hier, daß Grafikstufe 5 mit Textfenster verwendet wird. Außerdem will ich kurz auf den Schwierigkeitsgrad SG eingehen. Der Schwierigkeitsgrad SG bezeichnet die Suchtiefe, bis zu der der Spielbaum vom Computer generiert wird. Je größer also SG, desto besser spielt der Computer, desto größer aber auch die Denkzeit. Bemerkenswert ist, daß der Spieler die Suchtiefe frei wählen kann.
Der "Spielerzug" befindet sich in den Zeilen 3000 bis 3240. Eingegeben werden muß hier vom Spieler die Variable SP, welche die Spalte bezeichnet, in die der Spielstein geworfen wird. Die Steine des Spielers werden rot, die Steine des Computers werden blau gezeichnet

Der "Zug des Computers" beginnt in Zeile 4000 und endet mit Zeile 4550. Nach wiederholtem Probieren habe ich festgestellt, daß dieser Teil des Programmes mit Worten äußerst schwer zu beschreiben ist. Ich habe mich nach einigem Zögern deshalb entschlossen, diesen Teil nicht weiter zu erläutern. Das befriedigt Sie vieleicht nur

ungenügend, jedoch würde die genaue Beschreibung den Rahmen dieses Buches sprengen.

In den Zeilen 5000 bis 5410 wird getestet, ob durch Auffüllen des Feldes (SP,B(SP)-1) ein Vierer entstanden ist. Wenn ja, wird VIER gleich 1 gesetzt, ansonsten VIER gleich 0.

Interessant ist wieder das Unterprogramm "Stellung bewerten". Das Bewertungsprogramm ist ja in erster Linie für die Spielstärke des Computers verantwortlich. Das Prinzip dieses Bewertungsprogrammes besteht darin, das gesamte Spielfeld abzutasten. Liegt ein leeres Feld vor, so wird überprüft, ob durch Auffüllen dieses Feldes ein Vierer für den Computer oder den Spieler entsteht. Ein Vierer für den Computer ergibt +50 Punkte, ein Vierer für den Spieler ergibt -80 Punkte. Liegt ein besetztes Feld vor, so springt der Computer nach Zeile 6200, wo dieses Feld bewertet wird. Der Computer geht dabei von diesem Feld aus in sämtliche mögliche Richtungen. Berücksichtigt wird dann, ob ein Vierer in dieser Richtung überhaupt noch möglich ist und wieviele Steine des Vierers schon vorhanden sind.

Sie sehen, das Bewertungsprogramm ist ebenfalls äußerst komplex. Sie werden mich vieleicht fragen, wie man so ein Bewertungsprogramm erstellt. Die Antwort von mir lautet dann: "Hauptsächlich durch Probieren". Zuerst erstellt man eine lauffähige Version. Danach erkennt man beim direkten Spiel mit dem Computer die Stärken und Schwächen des Programmes und kann es entsprechend verbessern. Auf diese Weise ist auch das hier vorgestellte Bewertungsprogramm entstanden. Sie können ja einmal versuchen, eine eigene Version herzustellen, doch ich glaube kaum, daß dieses von mir erstellte Bewertungsprogramm noch grundlegend verbessert werden kann.
Tippen Sie das Programm einfach ein und spielen Sie damit. Ich glaube, daß Sie über die Spielstärke des Computers angenehm überrascht sein werden. Die hohe Denkzeit des Computers trübt jedoch etwas diesen positiven Eindruck.

PROGRAMMLAUF 1

VIER GEWINNT 1

```
Wer soll anfangen (S/C)? C
Welchen Schwierigkeitsgrad (1-5)? 1
Mein Zug: Spalte 4
Welche Spalte? 4
Mein Zug: Spalte 4
Welche Spalte? 4
Mein Zug: Spalte 2
Welche Spalte? 2
Mein Zug: Spalte 3
Welche Spalte? 1
Mein Zug: Spalte 5

Ich habe gewonnen!!

Noch ein Spiel (J/N)? J
```

VARIABLENLISTE 1

```
A(X,Y)        :Spielfeld (0=leeres Feld, 1=Spieler, 5=Computer)
B(X)          :Gibt das unterste leere Feld jeder Spalte an.
               B(2)=3 z.B. bedeutet, daß Feld (2,1) und
               Feld (2,2) besetzt sind und daß Feld (2,3)
               leer ist.
BEW(X)        :Gibt beim Generieren des Spielbaumes die
               Bewertung an, wobei das Argument die jeweilige
               Suchtiefe bezeichnet.
SPA(X),X(X) :Werden zum Generieren des Baumes benötigt.
A$            :Eingaqabestring der Länge Eins.
ZUG           :Gibt die Anzahl der gemachten Züge an.
```

H	:Indikator für am Zug befindlichen Spieler. (1=Spieler, 5=Computer).
SG	:Schwierigkeitsgrad; der Spielbaum wird bis zur Suchtiefe SG generiert.
VIER	:Variable, die anzeigt, ob ein Vierer vorliegt (Vier=1) oder nicht (Vier=0).
SP	:Gibt die Spalte an, in die der Spielstein geworfen wird.
TI	:Beim Generieren des Baumes zeigt diese Variable die Suchtiefe an.
BEW	:Die Bewertung, die eine Spielstellung mittels des Unterprogramms "Stellung bewerten" erhält.
S	:Bezeichnet den Y-Wert des zuletzt aufgefüllten Feldes. Wird beim Unterprogramm "Vierer verstanden? benötigt.
F	:Gibt beim Bewerten des Feldes (X,Y) den Inhalt dieses Feldes an (1 oder 5).
HVOR,BVOR	:Wird beim Bewerten zum Zwischenspeichern von H und B(X) benötigt.

Listing 1: Vier Gewinnt

```
1000 Rem +++ Initialisierung +++++++++++++++++++++++++++++++++++

1100 Rem --- Dimensionierung
1110 Dim A(7,6),B(7),A$(1),BEW(5),SPA(5),X(5)
1120 ZUG=0

1200 Rem --- Setzen der Felder
1210 For I=1 to 7: For J=1 to 6: A(I,J)=0: Next J: Next I
1220 For I=1 to 7: B(I)=1: Next I

2000 Rem +++ Hauptprogramm +++++++++++++++++++++++++++++++++++

2100 Rem --- Spielfeld zeichnen
2110 Graphics 5: Setcolor 0,3,4: Color 2
2120 For I=12 to 68 Step 8: Plot I,0: Drawto I,36: Next I
2130 For I=0 to 36 Step 6: Plot 12,I: Drawto 68,I: Next I

2200 Rem --- Eingabe, wer anfaengt
2210 Print " Vier gewinnt "
2220 Print
2230 Print "Wer soll anfangen (S/C) ";
2240 Input A$
2250 If A$="S" then H=1: Goto 2300
2260 If A$="C" then H=5: Goto 2300
2270 Goto 2230

2300 Rem --- Eingabe Schwierigkeitsgrad
2310 Print "Welchen Schwierigkeitsgrad (1-5) ";
2320 Input SG
2330 If SG<1 or SG>5 then 2310

2400 Rem --- Spielablauf
2410 ZUG=ZUG+1
2420 If H=1 then gosub 3000          :Rem Spielerzug
2430 If H=5 then gosub 4000          :Rem Computerzug
2440 Gosub 5000                      :Rem Spielende?
2450 If VIER=0 and ZUG<42 then H=6-H: Goto 2410

2500 Rem --- Spielende
2510 For I=1 to 3: For J=4 to 3 Step -1: For K=14 to 0 Step -1
2520 Sound 1,J+40,10,K: Sound 2,J+40+1,10,K
2530 Next K: Next J: Next I
2540 Print
```

```
2550 If ZUG=42 then print "Unentschieden!!"
2560 If ZUG<42 and H=1 then print "Sie haben gewonnen!!"
2570 If ZUG<42 and H=5 then print "Ich habe gewonnen!!"
2580 Print
2590 Print "Noch ein Spiel (J/N) ";
2600 Input A$
2610 If A$="J" then Run
2620 End

3000 Rem +++ Spielerzug ++++++++++++++++++++++++++++++++++++++++

3100 Rem --- Zugeingabe
3110 Print "Welche Spalte ";
3120 Input SP
3130 If SP<1 or SP>7 then 3110
3140 If B(SP)=7 then print "Diese Spalte ist voll!": Goto 3110

3200 Rem --- Zug ausfuehren
3210 Color 1
3220 For I=37-6*B(SP) to 41-6*B(SP): Plot 5+8*SP,I: Drawto 11+
     8*SP,I: Next I
3230 A(SP,B(SP))=1: B(SP)=B(SP)+1
3240 Return

4000 Rem +++ Zug des Computers +++++++++++++++++++++++++++++++++
4010 Print "Mein Zug: Spalte ";

4100 Rem --- Generieren des Baumes
4110 BEW(0)=1000: TI=1: If SG>43-ZUG then SG=43-ZUG
4120 BEW(TI)=(3-H)*500: X(TI)=0
4130 X(TI)=X(TI)+1: If X(TI)=8 then 4400
4140 If B(X(TI))=7 then 4130
4150 A(X(TI),B(X(TI)))=H: B(X(TI))=B(X(TI))+1
4160 SP=X(TI): Gosub 5000
4170 If VIER=1 then 4300
4180 If TI<SG then TI=TI+1: H=6-H: Goto 4120

4200 Rem --- Geforderte Suchtiefe erreicht
4210 Gosub 6000          :Rem Stellung bewerten
4220 If H=1 and BEW<BEW(TI) then BEW(TI)=BEW: SPA(TI)=X(TI)
4230 If H=5 and BEW>BEW(TI) then BEW(TI)=BEW: SPA(TI)=X(TI)
4240 B(X(TI))=B(X(TI))-1: A(X(TI),B(X(TI)))=0: Goto 4130
```

```
4300 Rem --- Abbruch, da Vierer vorliegt
4310 If H=1 and -500>BEW(TI-1) then BEW(TI-1)=-500: SPA(TI-1)=X(TI)
4320 If H=5 and 500<BEW(TI-1) then BEW(TI-1)=500: SPA(TI-1)=X(TI)
4330 B(X(TI))=B(X(TI))-1: A(X(TI),B(X(TI)))=0
4340 If TI=1 then 4500
4350 TI=TI-1: H=6-H
4360 B(X(TI))=B(X(TI))-1: A(X(TI),B(X(TI)))=0: Goto 4130

4400 Rem --- Alle Zuege in dieser Tiefe ausgefuehrt
4410 If H=1 and BEW(TI)>BEW(TI-1) then BEW(TI-1)=BEW(TI): SPA(
     TI-1)=X(TI-1)
4420 If H=5 and BEW(TI)<BEW(TI-1) then BEW(TI-1)=BEW(TI): SPA(
     TI-1)=X(TI-1)
4430 If TI>1 then TI=TI-1: H=6-H: B(X(TI))=B(X(TI))-1: A(X(TI),
     B(X(TI)))=0: Goto 4130
4440 SPA(0)=SPA(1)

4500 Rem --- Zug ausgeben und ausfuehren
4510 SP=SPA(0)
4520 Print SP: Color 3
4530 For I=37-6*B(SP) to 41-6*B(SP): Plot 5+8*SP,I: Drawto 11+
     8*SP,I: Next I
4540 A(SP,B(SP))=5: B(SP)=B(SP)+1
4550 Return

5000 Rem +++ Vierer entstanden? ++++++++++++++++++++++++++++++++
5010 VIER=1: S=B(SP)-1

5100 Rem --- Senkrecht
5110 If S<4 then 5200
5120 If A(SP,S-1)+A(SP,S-2)+A(SP,S-3)=3*H then Return

5200 Rem --- Waagrecht
5210 For I=4 to 7
5220 If A(I,S)+A(I-1,S)+A(I-2,S)+A(I-3,S)=4*H then Return
5230 Next I

5300 Rem --- Diagonal
5310 For I=4 to 6
5320 If SP+S-I<1 or SP+S-I>4 then 5340
5330 If A(SP+S-I,I)+A(SP+S-I+1,I-1)+A(SP+S-I+2,I-2)+A(SP+S-I+3,
     I-3)=4*H then Return
```

```
5340 If SP+I-S>7 or SP+I-S<4 then 5360
5350 If A(SP+I-S,I)+A(SP+I-S-1,I-1)+A(SP+I-S-2,I-2)+A(SP+I-S-3,
     I-3)=4*H then Return
5360 Next I

5400 Rem --- Kein Vierer vorhanden
5410 VIER=0: Return

6000 Rem +++ Stellung bewerten ++++++++++++++++++++++++++++++++
6010 BEW=0

6100 Rem --- Spielfeld abtasten
6110 For X=1 to 7: For Y=1 to 6
6120 F=A(X,Y): If F>0 then gosub 6200: Goto 6170
6130 SP=X: HVOR=H: BVOR=B(X): B(X)=Y+1
6140 H=1: A(SP,Y)=H: Gosub 5000: If VIER=1 then BEW=BEW-80
6150 H=5: A(SP,Y)=H: Gosub 5000: If VIER=1 then BEW=BEW+50
6160 A(SP,Y)=0: B(X)=BVOR: H=HVOR
6170 Next Y: Next X
6180 Return

6200 Rem --- Bewertung Senkrecht
6210 I=0: A=0
6220 I=I+1: If Y-I<1 then 6240
6230 If A(X,Y-I)=F then A=A+1: Goto 6220
6240 J=0
6250 J=J+1: If Y+J>6 or J=4 then 6280
6260 If A(X,Y+J)=0 then 6250
6270 If A(X,Y+J)=F then A=A+1: Goto 6250
6280 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)

6300 Rem --- Bewertung Waagrecht
6310 I=0: A=0
6320 I=I+1: If X-I<1 or I=4 then 6350
6330 If A(X-I,Y)=0 then 6320
6340 If A(X-I,Y)=F then A=A+1: Goto 6320
6350 J=0
6360 J=J+1: If X+J>7 or J=4 then 6390
6370 If A(X+J,Y)=0 then 6360
6380 If A(X+J,Y)=F then A=A+1: Goto 6360
6390 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
```

```
6400 Rem --- Bewertung Diagonal
6410 I=0: A=0
6420 I=I+1: If X-I<1 or Y-I<1 or I=4 then 6450
6430 If A(X-I,Y-I)=0 then 6420
6440 If A(X-I,Y-I)=F then A=A+1: Goto 6420
6450 J=0
6460 J=J+1: If X+J>7 or Y+J>6 or J=4 then 6490
6470 If A(X+J,Y+J)=0 then 6460
6480 If A(X+J,Y+J)=F then A=A+1: Goto 6460
6490 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
6500 I=0: A=0
6510 I=I+1: If X-I<1 or Y+I>6 or I=4 then 6540
6520 If A(X-I,Y+I)=0 then 6510
6530 If A(X-I,Y+I)=F then A=A+1: Goto 6510
6540 J=0
6550 J=J+1: If X+J>7 or Y-J<1 or J=4 then 6580
6560 If A(X+J,Y-J)=0 then 6550
6570 If A(X+J,Y-J)=F then A=A+1: Goto 6550
6580 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
6600 Return
```

VERBESSERUNG DES PROGRAMMES

Das erste, was uns beim Spiel mit dem Computer auffällt, ist die Tatsache, daß der Computer zu den Stellungen immer nur genau einen Zug auswählt. Das kommt daher, daß bei der Zugauswahl der Zufall völlig ausgeschlossen ist. Nach einigen Spielen erweist sich dies jedoch als ziemlich störend. Die Züge des Computers lassen sich mit der Zeit nämlich genau vorherbestimmen und der Computer wiederholt immer wieder die gleichen Partien. Um das Spiel interessanter zu gestalten, müssen wir bei der Zugauswahl des Computers den Zufall mit einbeziehen. Am besten wäre es, wenn die Größe des Zufalls wählbar ist. Das erreichen wir auf ganz einfache Weise. Wir fügen zusätzlich die Zeilen 2340 bis 2360 ein, in denen der Spieler den Zufallsfaktor ZF eingeben muß:

```
2340 Print "Welchen Zufallsfaktor (0-5) ";
2350 Input ZF
2360 If ZF^0 or ZF^5 then 2340
```

Nun ändern wir Zeile 6180 folgendermaßen ab:

```
6180 BEW=BEW+Int(Rnd(0)+10*ZF): Return
```

Bei ZF=0 hat sich nichts geändert. Bei ZF=1 jedoch wird zu der Bewertung jeder Stellung ein zufälliger Wert zwischen 0 und 9 addiert, bei ZF=2 ein zufälliger Wert zwischen 0 und 19 usw. Je größer also ZF, desto größer der Einfluß des Zufalls auf die Zugauswahl.

Nun zum schwierigeren Teil: Die Senkung der Denkzeit des Computers. Die Mittel hierfür lautet "Maschinensprache". Natürlich schreiben wir nicht das ganze Programm in Maschinensprache, das wäre viel zu kompliziert. Wir ersetzen vielmehr lediglich einen Teil des Programmes, welcher großen Einfluß auf die Denkzeit des Computers hat. Meine Wahl fiel auf das Unterprogramm "Vierer entstanden?".

Zur Vereinfachung des Maschinen-Programmes speichern wir das Spielfeld zusätzlich in den Speicheradressen 1540 bis 1581 ab. Das Maschinensprache-Unterprogramm selbst legen wir in einen freien Speicherbereich, den wir mit dem String MSs reservieren. Die Variable MSANF gibt die Anfangsadresse dieses Speicherbereichs an.

In Zeile 5010 wird das Maschinensprache-Unterprogramm aufgerufen. Dabei werden an dieses Programm drei Werte übergeben. Es soll ja getestet werden, ob durch Auffüllen des Feldes (SP,B(SP)-1) ein Vierer entstanden ist. Nummerieren wir die Felder des Spielfeldes auf folgende Weise:

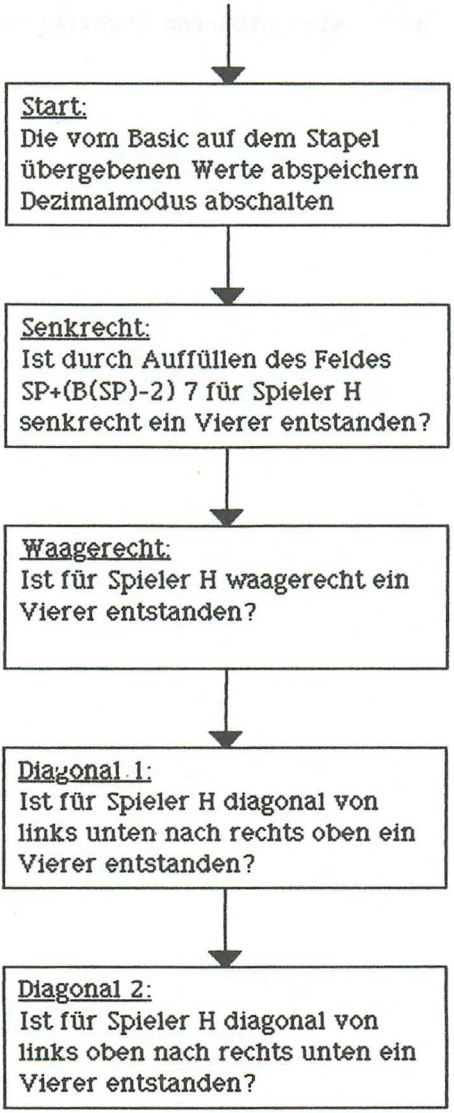
36	37	38	39	40	41	42
29	30	31	32	33	34	35
22	23	24	25	26	27	28
15	16	17	18	19	20	21
8	9	10	11	12	13	14
1	2	3	4	5	6	7

Dem Feld (SP,B(SP)-1) entspricht die Nummer SP+(B(SP)-2)7. Dieser Wert wird übergeben, außerdem die Variable SP und H.

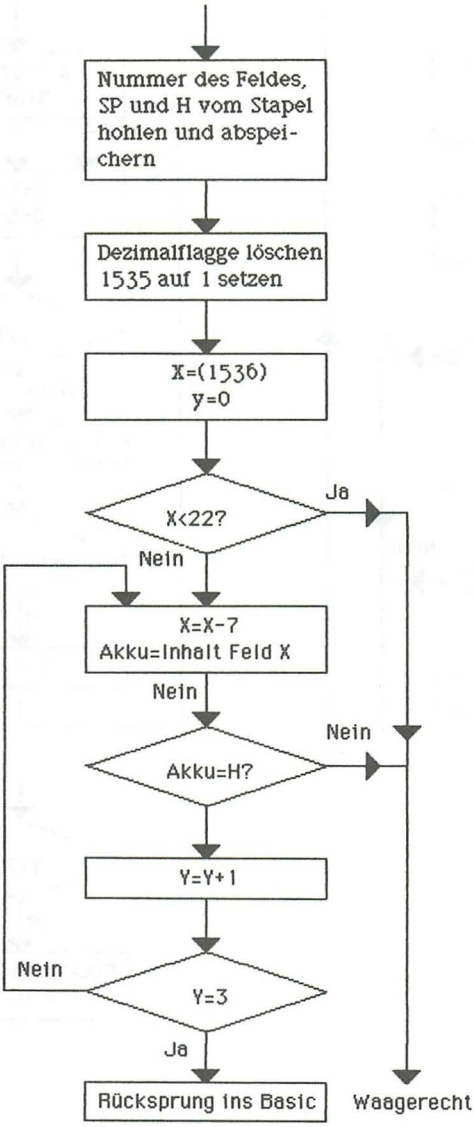
Das Flußdiagramm und das Assembler-Listing finden Sie auf den nächsten Seiten abgebildet. Das Flußdiagramm zu dem Teil "Diagonal 2" fehlt, da dieser Teil nahezu identisch ist mit dem Teil "Diagonal 1".

Die Schnelligkeit des Programmes ließe sich noch erheblich steigern, wenn weitere Teile des "Computerzuges" in Maschinensprache abgefaßt werden würden. Vielleicht gelingt es einem Leser, den gesamten in Zeile 6000 beginnenden Programmteil "Stellung bewerten" durch Maschinensprache zu ersetzen! Die Denkzeit des Computers wäre dann erheblich kürzer, so daß auch mit größeren Suchtiefen gearbeitet werden könnte.

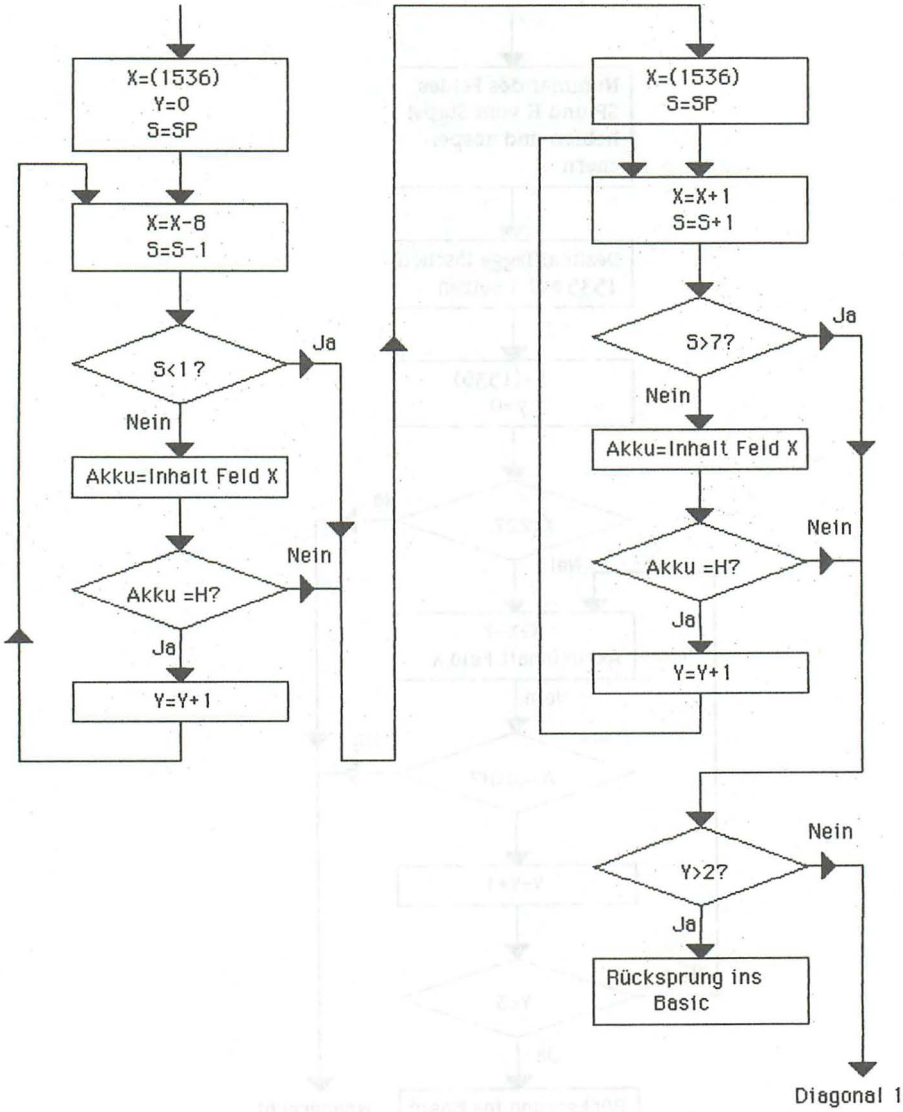
Flußdiagramm Übersicht



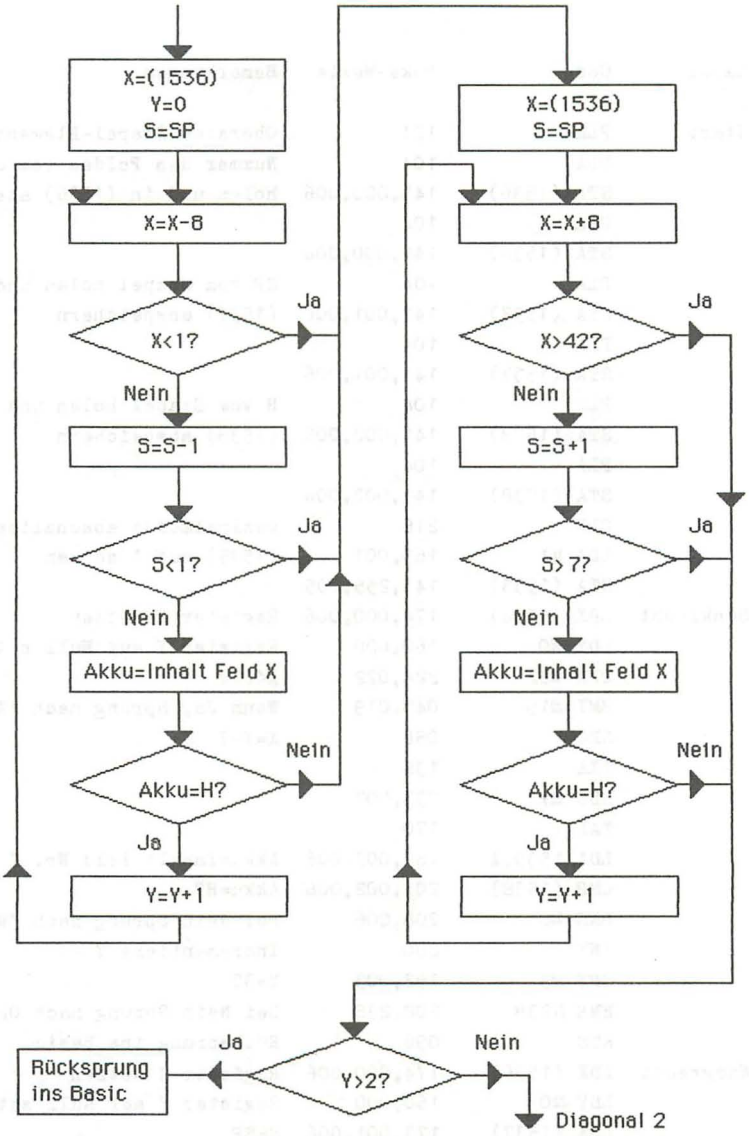
Flußdiagramm 'Start' und 'Senkrecht'



Flußdiagramm 'Waagerecht'



Flußdiagramm 'Diagonal 1'



Assembler-Listing

Befehl	Label	Code	Poke-Werte	Bemerkungen
001	Start	PLA	104	Oberstes Stapel-Element holen
002		PLA	104	Nummer des Feldes vom Stapel
003		STA (1536)	141,000,006	holen und in (1536) abspeichern
004		PLA	104	
005		STA (1536)	141,000,006	
006		PLA	104	SP vom Stapel holen und in
007		STA (1537)	141,001,006	(1537) abspeichern
008		PLA	104	
009		STA (1537)	141,001,006	
010		PLA	104	H vom Stapel holen und in
011		STA (1538)	141,002,006	(1538) abspeichern
012		PLA	104	
013		STA (1538)	141,002,006	
014		CLD	216	Dezimalmodus abschalten
015		LDA #1	169,001	(1535) auf 1 setzen
016		STA (1535)	141,255,005	
017	Senkrecht	LDX (1536)	174,000,006	Register X setzen
018		LDY #0	160,000	Register Y auf Null setzen
019		CPX #22	224,022	X<22?
020		BMI #19	048,019	Wenn Ja, Sprung nach 'Waagrecht'
021		SEC	056	X=X-7
022		TXA	138	
023		SBC #7	233,007	
024		TAX	170	
025		LDA 1539,X	189,003,006	Akku=Inhalt Feld Nr. X
026		CMP (1538)	205,002,006	Akku=H?
027		BNE #6	208,006	Bei Nein Sprung nach 'Waagrecht'
028		INY	200	Incrementiere Y
029		CPY #3	192,003	Y=3?
030		BNE #238	208,238	Bei Nein Sprung nach 021
031		RTS	096	Rücksprung ins Basic
032	Waagrecht	LDX (1536)	174,000,006	Register X setzen
033		LDY #0	160,000	Register Y auf Null setzen
034		LDA (1537)	173,001,006	S=SP

```
035                 STA (1539)    141,003,006
036                 DEX           202          Decrementiere X
037                 DEC (1539)    206,003,006  S=S-1
038                 LDA (1539)    173,003,006  S<1?
039                 CMP #1        201,001
040                 BMI #13       048,013      Wenn Ja, Sprung nach 047
041                 LDA 1539,X    189,003,006  Akku=Inhalt Feld Nr. X
042                 CMP (1538)    205,002,006  Akku=H?
043                 BNE #5        208,005      Bei Nein Sprung nach 047
044                 INY           200          Y=Y+1
045                 CPY #0        192,000      Sprung nach 036
046                 BPL #232      016,232
047                 LDX (1536)    174,000,006  X=(1536)
048                 LDA (1537)    173,001,006  S=SP
049                 STA (1539)    141,003,006
050                 INX           232          X=X+1
051                 INC (1539)    238,003,006  S=S+1
052                 LDA (1539)    173,003,006  S>7?
053                 CMP #8        201,008
054                 BPL #13       016,013      Wenn Ja, Sprung nach 061
055                 LDA 1539,X    189,003,006  Akku=Inhalt Feld Nr. X
056                 CMP (1538)    205,002,006  Akku=H?
057                 BNE #5        208,005      Bei Nein Sprung nach 061
058                 INY           200          Y=Y+1
059                 CPY #0        192,000      Sprung nach 050
060                 BPL #232      016,232
061                 CPY #3        192,003      Y>2?
062                 BMI #1        048,001      Bei Nein Sprung zu 'Diagonal 1'
063                 RTS           096          Rücksprung ins Basic
064   Diagonal 1    LDX (1536)    174,000,006  X=(1536)
065                 LDY #0        160,000      Y=0
066                 LDA (1537)    173,001,006  S=SP
067                 STA (1539)    141,003,006
068                 SEC           056          X=X-8
069                 TXA           138
070                 SBC #8        233,008
071                 TAX           170
072                 CPX #1        224,001      X<1?
073                 BMI #23       048,023      Wenn Ja, Sprung nach 084
```

```
074                DEC (1539)    206,003,006  S=S-1
075                LDA (1539)    173,003,006  S<1?
076                CMP #1        201,001
077                BMI #13       048,013      Wenn Ja, Sprung nach 084
078                LDA 1539,X    189,003,006  Akku=Inhalt Feld Nr. X
079                CMP (1538)    205,002,006  Akku=H?
080                BNE #5        208,005      Bei Nein Sprung nach 084
081                INY           200          Y=Y+1
082                CPY #0        192,000      Sprung nach 068
083                BPL #224      016,224
084                LDX (1536)    174,000,006  X=(1536)
085                LDA (1537)    173,001,006  S=SP
086                STA (1539)    141,003,006
087                CLC           024          X=X+8
088                TXA           138
089                ADC #8        105,008
090                TAX           170
091                CPX #43       224,043      X>42?
092                BPL #23       016,023      Wenn Ja, Sprung nach 103
093                INC (1539)    238,003,006  S=S+1
094                LDA (1539)    173,003,006  S>7?
095                CMP #8        201,008
096                BPL #13       016,013      Wenn Ja, Sprung nach 103
097                LDA 1539,X    189,003,006  Akku=Inhalt Feld Nr. X
098                CMP (1538)    205,002,006  Akku=H?
099                BNE #5        208,005      Bei Nein Sprung nach 103
100                INY           200          Y=Y+1
101                CPY #0        192,000      Sprung nach 087
102                BPL #224      016,224
103                CPY #3        192,003      Y>2?
104                BMI #1        048,001      Bei Nein Sprung zu 'Diagonal 2'
105                RTS           096          Rücksprung ins Basic
106  Diagonal 2    LDX (1536)    174,000,006  X=(1536)
107                LDY #0        160,000      Y=0
108                LDA (1537)    173,001,006  S=SP
109                STA (1539)    141,003,006
110                SEC           056          X=X-6
111                TXA           138
112                SBC #6        233,006
113                TAX           170
```

```
114   CPX #1        224,001       X<1?
115   BMI #23       048,023       Wenn Ja, Sprung nach 126
116   INC (1539)    238,003,006   S=S+1
117   LDA (1539)    173,003,006   S>7?
118   CMP #8        201,008
119   BPL #13       016,013       Wenn Ja, Sprung nach 126
120   LDA 1539,X    189,003,006   Akku=Inhalt Feld Nr.X
121   CMP (1538)    205,002,006   Akku=H?
122   BNE #5        208,005       Bei Nein Sprung nach 126
123   INY           200           Y=Y+1
124   CPY #0        192,000       Sprung nach 110
125   BPL #224      016,224
126   LDX (1536)    174,000,006   X=(1536)
127   LDA (1537)    173,001,006   S=SP
128   STA (1539)    141,003,006
129   CLC           024           X=X+6
130   TXA           138
131   ADC #6        105,006
132   TAX           170
133   CPX #43       224,043       X>42?
134   BPL #23       016,023       Wenn Ja, Sprung nach 145
135   DEC (1539)    206,003,006   S=S-1
136   LDA (1539)    173,003,006   S<1?
137   CMP #1        201,001
138   BMI #13       048,013       Wenn Ja, Sprung nach 145
139   LDA 1539,X    189,003,006   Akku=Inhalt Feld Nr. X
140   CMP (1538)    205,002,006   Akku=H?
141   BNE #5        208,005       Bei Nein Sprung nach 145
142   INY           200           Y=Y+1
143   CPY #0        192,000       Sprung nach 129
144   BPL #224      016,224
145   CPY #3        192,003       Y>2?
146   BMI #1        048,001       Bei Nein Sprung nach 148
147   RTS           096           Rücksprung ins Basic
148   LDA #0        169,000       (1535)=0
149   STA (1535)    141,255,005
150   RTS           096           Rücksprung ins Basic
```

PROGRAMMLAUF 2

VIER GEWINNT 1

```
Wer soll anfangen (S/C)? C
Welchen Schwierigkeitsgrad (1-5)? 2
Welchen Zufallsfaktor (0-5)? 1
Mein Zug: Spalte 6
Welche Spalte? 4
Mein Zug: Spalte 4
Welche Spalte? 4
Mein Zug: Spalte 4
Welche Spalte? 2
Mein Zug: Spalte 1
Welche Spalte? 2
Mein Zug: Spalte 2
Welche Spalte? 4
Mein Zug: Spalte 5
Welche Spalte? 5
Mein Zug: Spalte 1
Welche Spalte? 5
Mein Zug: Spalte 5
Welche Spalte? 1
Mein Zug: Spalte 2
Welche Spalte? 5
Mein Zug: Spalte 2
Welche Spalte? 2
Mein Zug: Spalte 4
Welche Spalte? 3
Mein Zug: Spalte 1
Welche Spalte? 1
Mein Zug: Spalte 5
Welche Spalte? 1
```

```
Mein Zug: Spalte 7
Welche Spalte? 6
Mein Zug: Spalte 6
Welche Spalte? 6
Mein Zug: Spalte 6
Welche Spalte? 6
Mein Zug: Spalte 7
Welche Spalte? 7
Mein Zug: Spalte 7
Welche Spalte? 7
Mein Zug: Spalte 7
Welche Spalte? 3
Mein Zug: Spalte 3

Ich habe gewonnen!!

Noch ein Spiel (J/N)? J
```

VARIABLENLISTE 2

A(X,Y)	:Spielfeld
B(X)	:Unterstes leeres Feld jeder Spalte
BEW(X)	:Bewertung, wobei das Argument die Suchtiefe angibt.
SPA(X),X(X)	:Generieren des Spielbaumes
A$	:Eingabestring der Länge 1
MS$	:String der Länge 316. Dieser String reserviert den notwendigen Speicherplatz für das Maschinensprache-Unterprogramm.
ZUG	:Anzahl gemachter Züge
MSANF	:Anfangsadresse des Maschinensprache-Unterprogramms
H	:Indikator für am Zug befindlichen Spieler
SG	:Schwierigkeitsgrad
ZF	:Zufallsfaktor. Je größer diese Variable, desto größer ist der Einfluß des Zufalls auf die Zugauswahl des Computers.
VIER	:Zeigt an, ob ein Vierer vorliegt
SP	:Gibt beim Zug die Spalte an
TI	:Bearbeitende Suchtiefe
BEW	:Bewertung einer Spielstellung
HVOR,BVOR	:Zwischenspeichern von H und B(X)

Listing 2: Vier Gewinnt

```
1000 Rem +++ Initialisierung +++++++++++++++++++++++++++++++++++

1100 Rem --- Dimensionierung
1110 Dim A(7,6),B(7),A$(1),BEW(5),SPA(5),X(5),MS$(316)
1120 ZUG=0: MSANF=ADR(MS$)

1200 Rem --- Setzen der Felder
1210 For I=1 to 7: For J=1 to 6: A(I,J)=0: Next J: Next I
1220 For I=1 to 7: B(I)=1: Next I
1230 For I=1540 to 1581: Poke I,0: Next I

1300 Rem --- Einlesen Maschinenprogramm
1310 For I=0 to 315
1320 Read J: Poke MSANF+I,J
1330 Next I

2000 Rem +++ Hauptprogramm +++++++++++++++++++++++++++++++++++

2100 Rem --- Spielfeld zeichnen
2110 Graphics 5: Setcolor 0,3,4: Color 2
2120 For I=12 to 68 Step 8: Plot I,0: Drawto I,36: Next I
2130 For I=0 to 36 Step 6: Plot 12,I: Drawto 68,I: Next I

2200 Rem --- Eingabe, wer anfaengt
2210 Print "Vier Gewinnt"
2220 Print
2230 Print "Wer soll anfangen (S/C) ";
2240 Input A$
2250 If A$="S" then H=1: Goto 2300
2260 If A$="C" then H=5: Goto 2300
2270 Goto 2230

2300 Rem --- Eingabe Schwierigkeitsgrad,Zufallsfaktor
2310 Print "Welchen Schwierigkeitsgrad (1-5) ";
2320 Input SG
2330 If SG<1 or SG>5 then 2310
2340 Print "Welchen Zufallsfaktor (0-5) ";
2350 Input ZF
2360 If ZF<0 or ZF>5 then 2340
```

```
2400 Rem --- Spielablauf
2410 ZUG=ZUG+1
2420 If H=1 then gosub 3000      :Rem Spielerzug
2430 If H=5 then gosub 4000      :Rem Computerzug
2440 Gosub 5000                  :Rem Spielende?
2450 If VIER=0 and ZUG<42 then H=6-H: Goto 2410

2500 Rem --- Spielende
2510 For I=1 to 3: For J=4 to 3 Step -1: For K=14 to 0 Step -1
2520 Sound 1,J+40,10,K: Sound 2,J+40+1,10,K
2530 Next K: Next J: Next I
2540 Print
2550 If ZUG=42 then print "Unentschieden!!"
2560 If ZUG<42 and H=1 then print "Sie haben gewonnen!!"
2570 If ZUG<42 and H=5 then print "Ich habe gewonnen!!"
2580 Print
2590 Print "Noch ein Spiel (J,N) ";
2600 Input A$
2610 If A$="J" then Run
2620 End

3000 Rem +++ Spielerzug ++++++++++++++++++++++++++++++++++++++++

3100 Rem --- Zugeingabe
3110 Print "Welche Spalte ";
3120 Input SP
3130 If SP<1 or SP>7 then 3110
3140 If B(SP)=7 then print "Diese Spalte ist voll!": Goto 3110

3200 Rem --- Zug ausfuehren
3210 Color 1
3220 For I=37-6*B(SP) to 41-6*B(SP): Plot 5+8*SP,I: Drawto 11+
     8*SP,I: Next I
3230 A(SP,B(SP))=1: Poke 1539+SP+(B(SP)-1)*7,1: B(SP)=B(SP)+1
3240 Return

4000 Rem +++ Zug des Computers ++++++++++++++++++++++++++++++++
4010 Print "Mein Zug: Spalte ";

4100 Rem --- Generieren des Baumes
4110 BEW(0)=1000: TI=1: If SG>43-ZUG then SG=43-ZUG
4120 BEW(TI)=(3-H)*500: X(TI)=0
4130 X(TI)=X(TI)+1: If X(TI)=8 then 4400
4140 If B(X(TI))=7 then 4130
```

```
4150 A(X(TI),B(X(TI)))=H: Poke 1539+X(TI)+(B(X(TI))-1)*7,H:
     B(X(TI))=B(X(TI))+1
4160 SP=X(TI): Gosub 5000
4170 If VIER=1 then 4300
4180 If TI<SG then TI=TI+1: H=6-H: Goto 4120

4200 Rem --- Geforderte Suchtiefe erreicht
4210 Gosub 6000            :Rem Stellung bewerten
4220 If H=1 and BEW<BEW(TI) then BEW(TI)=BEW: SPA(TI)=X(TI)
4230 If H=5 and BEW>BEW(TI) then BEW(TI)=BEW: SPA(TI)=X(TI)
4240 If (H=1 and BEW<BEW(TI-1)) or (H=5 and BEW>BEW(TI-1))
     then 4330
4250 B(X(TI))=B(X(TI))-1: A(X(TI),B(X(TI)))=0: Poke 1539+X(TI)+
     (B(X(TI))-1)*7,0: Goto 4130

4300 Rem --- Abbruch, da Vierer vorliegt
4310 If H=1 and -500>BEW(TI-1) then BEW(TI-1)=-500: SPA(TI-1)=X(TI)
4320 If H=5 and 500<BEW(TI-1) then BEW(TI-1)=500: SPA(TI-1)=X(TI)
4330 B(X(TI))=B(X(TI))-1: A(X(TI),B(X(TI)))=0: Poke 1539+X(TI)+
     (B(X(TI))-1)*7,0
4340 If TI=1 then 4500
4350 TI=TI-1: H=6-H
4360 B(X(TI))=B(X(TI))-1: A(X(TI),B(X(TI)))=0: Poke 1539+X(TI)+
     (B(X(TI))-1)*7,0: Goto 4130

4400 Rem --- Alle Zuege in dieser Tiefe ausgefuehrt
4410 If H=1 and BEW(TI)>BEW(TI-1) then BEW(TI-1)=BEW(TI): SPA(
     TI-1)=X(TI-1)
4420 If H=5 and BEW(TI)<BEW(TI-1) then BEW(TI-1)=BEW(TI): SPA(
     TI-1)=X(TI-1)
4430 If TI>1 then TI=TI-1: H=6-H: B(X(TI))=B(X(TI))-1: A(X(TI),
     B(X(TI)))=0: Poke 1539+X(TI)+(B(X(TI))-1)*7,0: Goto 4130
4440 SPA(0)=SPA(1)

4500 Rem --- Zug ausgeben und ausfuehren
4510 SP=SPA(0)
4520 Print SP: Color 3
4530 For I=37-6*B(SP) to 41-6*B(SP): Plot 5+8*SP,I: Drawto 11+
     8*SP,I: Next I
4540 A(SP,B(SP))=5: Poke 1539+SP+(B(SP)-1)*7,5: B(SP)=B(SP)+1
4550 Return
```

```
5000 Rem +++ Vierer entstanden? ++++++++++++++++++++++++++++++
5010 Z=USR(MSANF,SP+(B(SP)-2)*7,SP,H)
5020 VIER=Peek(1535): Return

6000 Rem +++ Stellung bewerten ++++++++++++++++++++++++++++++
6010 BEW=0
6100 Rem --- Spielfeld abtasten
6110 For X=1 to 7: For Y=1 to 6
6120 F=A(X,Y): If F>0 then gosub 6200: Goto 6170
6130 SP=X: HVOR=H: BVOR=B(X): B(X)=Y+1
6140 H=1: Gosub 5000: If VIER=1 then BEW=BEW-80
6150 H=5: Gosub 5000: If VIER=1 then BEW=BEW+50
6160 B(X)=BVOR: H=HVOR
6170 Next Y: Next X
6180 BEW=BEW+Int(Rnd(0)*10*ZF): Return
6200 Rem --- Bewertung Senkrecht
6210 I=0: A=0
6220 I=I+1: If Y-I<1 then 6240
6230 If A(X,Y-1)=F then A=A+1: Goto 6220
6240 J=0
6250 J=J+1: If Y+J>6 or J=4 then 6280
6260 If A(X,Y+J)=0 then 6250
6270 If A(X,Y+J)=F then A=A+1: Goto 6250
6280 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
6300 Rem --- Bewertung Waagrecht
6310 I=0: A=0
6320 I=I+1: If X-I<1 or I=4 then 6350
6330 If A(X-I,Y)=0 then 6320
6340 If A(X-I,Y)=F then A=A+1: Goto 6320
6350 J=0
6360 J=J+1: If X+J>7 or J=4 then 6390
6370 If A(X+J,Y)=0 then 6360
6380 If A(X+J,Y)=F then A=A+1: Goto 6360
6390 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
6400 Rem --- Bewertung Diagonal
6410 I=0: A=0
6420 I=I+1: If X-I<1 or Y-I<1 or I=4 then 6450
6430 If A(X-I,Y-I)=0 then 6420
```

```
6440 If A(X-I,Y-I)=F then A=A+1: Goto 6420
6450 J=0
6460 J=J+1: If X+J>7 or Y+J>6 or J=4 then 6490
6470 If A(X+J,Y+J)=0 then 6460
6480 If A(X+J,Y+J)=F then A=A+1: Goto 6460
6490 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
6500 I=0: A=0
6510 I=I+1: If X-I<1 or Y+I>6 or I=4 then 6540
6520 If A(X-I,Y+I)=0 then 6510
6530 If A(X-I,Y+I)=F then A=A+1: Goto 6510
6540 J=0
6550 J=J+1: If X+J>7 or Y-J<1 or J=4 then 6580
6560 If A(X+J,Y-J)=0 then 6550
6570 If A(X+J,Y-J)=F then A=A+1: Goto 6550
6580 If I+J>4 then BEW=BEW+(F-4)*(2*A+I+J-4)
6600 Return

7000 Rem +++ Datas fuer Maschinenprogramm 'Vierer entstanden?' +++
7005 Data 104,104,141,0,6,104,141,0,6,104,141,1,6,104,141,1,6,
     104,141,2,6,104,141,2,6
7010 Data 216,169,1,141,255,5,174,0,6,160
7020 Data 0,224,22,48,19,56,138,233,7,170
7030 Data 189,3,6,205,2,6,208,6,200,192
7040 Data 3,208,238,96
7050 Data 174,0,6,160,0,173,1,6,141,3,6
7060 Data 202,206,3,6,173,3,6,201,1,48,13
7070 Data 189,3,6,205,2,6,208,5,200,192,0
7080 Data 16,232,174,0,6,173,1,6,141,3,6,232,238,3,6,173,3,6,201,
     8,16,13,189,3,6
7090 Data 205,2,6,208,5,200,192,0,16,232,192,3,48,1,96
7100 Data 174,0,6,160,0,173,1,6,141,3,6,56,138,233,8
7110 Data 170,224,1,48,23,206,3,6,173,3,6,201,1,48,13
7120 Data 189,3,6,205,2,6,208,5,200,192,0,16,224,174,0,6,173,1,
     6,141,3,6
7130 Data 24,138,105,8,170,224,43,16,23,238,3,6
7140 Data 173,3,6,201,8,16,13,189,3,6,205,2,6,208,5
7150 Data 200,192,0,16,224,192,3,48,1,96
7160 Data 174,0,6,160,0,173,1,6,141,3,6,56,138,233,6,170,224,1,
     48,23
```

```
7170 Data 238,3,6,173,3,6,201,8,16,13,189,3,6,205,2,6,208,5,200,
     192,0
7180 Data 16,224,174,0,6,173,1,6,141,3,6,24,138,105,6,170,224,43,
     16,23
7190 Data 206,3,6,173,3,6,201,1,48,13,189,3,6,205,2,6,208,5,200,
     192,0,16,224,192,3,48,1,96
7200 Data 169,0,141,255,5,96
```

LISTE DER VERWENDETEN BEFEHLE

Abs(X) -Berechnet den Absolutbetrag (entfernt das Minuszeichen).
Beispiele:
Print Abs(-9.5) 9.5
Print Abs(9.5) 9.5

AND -Logisches "Und". Das Ergebnis ist dann wahr, wenn beide durch "And" verknüpften Aussagen wahr sind.

Color -Ordnet einem bestimmten Farbregister eine Farbe zu. Hinter Color muß eine Zahl bzw. eine Variable zwischen 0 und 255 folgen.

Data/Read/Restore -Mit dem Befehl "Data" wird eine sogenannte Data-Liste angelegt. Diese kann aus Zahlen und Zeichen (Strings) bestehen, welche durch Kommas getrennt werden müssen.
Beispiel:
Data 10,Auto,Haus,7,3,8
Der Befehl "Read" liest nacheinander die Werte oder Zeichen aus den Data-Zeilen und weist sie der Variablen bzw. den durch Kommas getrennten Variablen hinter "Read" zu. Dabei wird erwartet, daß die Variablenart hinter Read den zu lesenden Werten entspricht. Der Read/Data-Zeiger wird dabei jedesmal um eins weitergezählt.
"Restore" setzt den Read/Data-Zeiger auf den ersten Wert in der ersten Data-Zeile des Programms. Folgt nach "Restore" noch eine Zeilennummer, so wird der Read/Data-Zeiger

auf den ersten Wert in dieser Zeile bzw. in der nächst folgenden Data-Zeile gestellt. Dadurch kann eine Data-Zeile mehrfach genutzt werden. Außerdem ist ein schnelles Auffinden von Daten in einer Data-Liste möglich.

Dim -Dient zum Anlegen von Tabellen, indem es für einfach und doppelt indizierte Variablen den notwendigen Speicherplatz reserviert. Wird ein String benutzt, so muß die Länge dieses Strings vorher mit diesem Befehl angegeben werden. Der Befehl kann mehrere Variablen gleichzeitig dimensionieren, falls diese durch Kommas getrennt sind. Der Versuch, eine schon vorher dimensionierte Variable nochmals zu dimensionieren führt zu einer Fehlermeldung.

Beispiele:

a) Dim A$(5) Stringvariable mit 5 Zeichen

b) Dim A(7) Eindimensionale Liste mit acht Elementen

c) Dim A(9,4) Zweidimensionale Liste mit 10 Zeilen und 5 Spalten

Drawto -Zeichnet von der augenblicklichen Grafik-Cursorposition aus eine Linie zu der hinter Drawto angegebenen Position. Hinter "Drawto" muß die X- und Y-Koordinate (durch Komma getrennt) folgen. Als Nullpunkt für die Positionsangaben wird die linke obere Bildschirmecke verwendet.

Beispiel:

Drawto 70,40

End -Beendet die Programmausführung und schaltet den Bildschirm in den Grafikmodus 0.

For/Next/Step -Programmierung einer Schleife ("Loop").

Beispiel:

For X=3 to 11 Step 2: Print X: Next X

Der vollständige Befehl beinhaltet Anfangswert des Schleifenzählers (For X=3), Endwert des Schleifenzählers (to 11) und Schrittweite (Step 2). Diese drei Werte können direkt oder als Variablen eingegeben werden. Die Schrittweite kann sowohl positiv als auch negativ sein. Jede Schleife wird mindestens einmal durchlaufen.

Gosub/Return -Aufruf eines in Basic geschriebenen Unterprogrammes und Rückkehr ins aufrufende Basic-Programm.

Beispiel:

```
Gosub 1000
A=1000: Gosub A
```

Goto -"Unbedingter Sprung". Springt zu der hinter "Goto" stehenden Zeilennummer und führt das Programm an dieser Stelle fort.

Beispiel:

```
Goto 200
A=200: Goto A
Goto 100+2*50
```

Graphics -Dient zum Auswählen eines Grafikmodi.

Graphics	Textfenster Zeilen	Textfenster Spalten	Grafikfenster Zeilen	Grafikfenster Spalten	Anzahl Farben	Hellig-Keiten pro Farbe	Speicher-bedarf Bild-schirm
0	24	40	-	-	2	8	993
1	4	40	20	20	5	8	513
1+16	-	-	24	20	5	8	513
2	4	40	10	20	5	8	261
2+16	-	-	12	20	5	8	261
3	4	40	20	40	4	8	273
3+16	-	-	24	40	4	8	273
4	4	40	40	80	2	8	537
4+16	-	-	48	80	2	8	537
5	4	40	40	80	4	8	1017
5+16	-	-	48	80	4	8	1017
6	4	40	80	160	2	8	2025
6+16	-	-	96	160	2	8	2025
7	4	40	80	160	4	8	3945
7+16	-	-	96	160	4	8	3945
8	4	40	160	320	2	8	7900
8+16	-	-	192	320	2	8	7900

If/then -Fortsetzung des Programmes an der hinter "then" bezeichneten Stelle, wenn die gestellte Bedingung zwischen "if" und "then" erfüllt ist, ansonsten Fortsetzung in nächster Zeile.

Input -Eingabe von Text oder Zahlen von der Tastatur. Es können auch mehrere durch Kommas getrennte Daten gleichzeitig eingegeben werden. Bei der Abarbeitung des Befehls wird ein Fragezeichen ausgegeben und der Computer wartet, bis die Eingabe mit "Return" abgeschlossen wird.
Beispiele:
Input A
Input A,B,C$,D

Int -Berechnet die größte ganze Zahl, die kleiner oder gleich dem Argument ist.
Beispiele:
Print Int(3.5) 3
Print Int(-2.3 -3
Print Int(-11) -11

Or -Logisches "Oder". Ergebnis ist dann wahr, wenn mindestens eine der beiden durch "or" verknüpften Aussagen wahr ist.

Peek -Liest den Inhalt einer Speicherstelle aus dem RAM- oder ROM-Bereich. Das Ergebnis ist eine ganze Zahl zwischen 0 und 255.

Plot -Der Befehl zeichnet einen Punkt oder ein durch Color definiertes Zeichen an der hinter "Plot" angegebenen Stelle im Grafikfenster.
Beispiel:
Plot 10,36

Poke -Dient zum Verändern von Speicherstellen im RAM-Bereich. Die Adresse muß zwischen 0 und 65535 liegen, der zu schreibende Wert zwischen 0 und 255.

Beispiel:

```
Poke 10500,23
Poke A,B
```

Print -Ausgabe von Informationen auf den Bildschirm.

Beispiele:

```
Print A
Print B$
Print "10",A,B$,B
```

Print "^" -Löschen des Textfensters.

Rem -Programm-Kommentar, wird vom Basic nicht weiter beachtet.

Rnd(0) -Erzeugt eine Zufallszahl zwischen 0 und 1 (immer kleiner 1).

Run -Startet ein Basic-Programm bei der niedrigsten Zeilennummer.

Setcolor -Legt Farbe und Helligkeit in einem Farbregister fest. Die erste Zahl bezeichnet den Farbeimer, die zweite die Farbe und die dritte die Helligkeit.

Sound -Steuert die vier Tongeneratoren des Atari an. Erwartet werden vier Zahlen: Sound A,B,C,D

A Nummer des Tongenerators (0-3)
B Tonhöhe (0=hoch, 255=tief)
C Verzerrungsgrad
D Lautstärke (0-16)

ANHANG

LITERATURVERZEICHNIS

MEIROVITZ/JACOBS: Spielschule des Denkens

BAUMANN, Rüdiger: Computerspiele und Knobeleien programmiert in Basic

AHL, David: Basic Computer Games

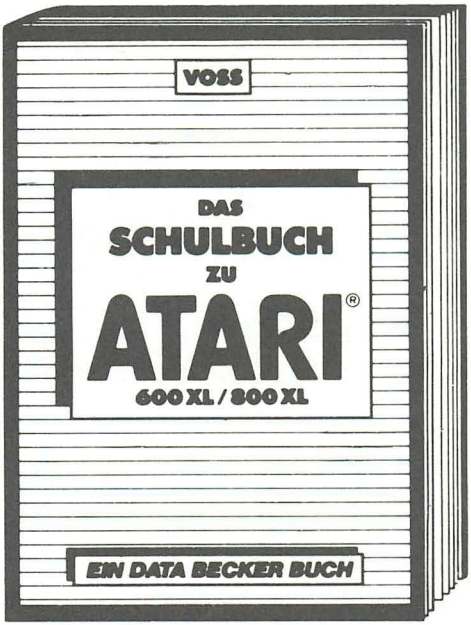
AHL, David: More Basic Computer Games

GARDNER, Martin: Mathematisches Labyrinth

GARDNER, Martin: Logik unterm Galgen

STEDE, Manfred: Einführung in die Künstliche Intelligenz, Bd.1

SCHUMNY, Harald: Ausgewählte Basic-Computerspiele (Atari 800)



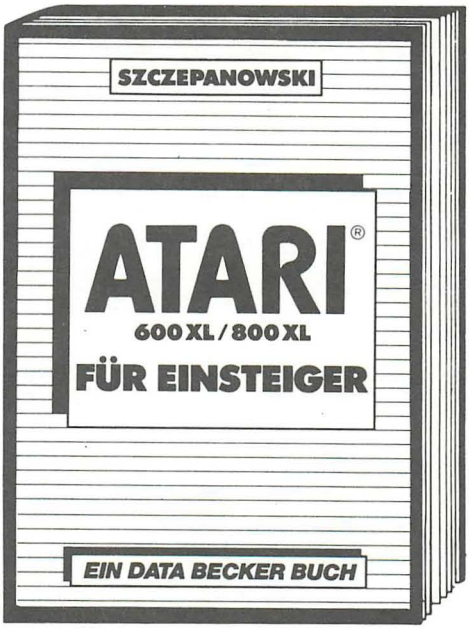
Interessant für Schüler, Lehrer und Eltern ist das Schulbuch zu ATARI 600/800XL. Vom Vokabelnlernen über Molekülbildung, exponentielles Wachstum bis zum Pythagoras und Geschichtszahlen enthält es – didaktisch gut aufbereitet – viele interessante Programme. Vor allem Schüler der Mittel- und Oberstufe werden in Mathe, Bio, Physik, Chemie, Sprachen und anderen Fächern wieder fit.

DAS SCHULBUCH ZU ATARI 600/800XL, 1984, über 300 Seiten, DM 49,–

Die neuen DATA BECKER BÜCHER

Interessant für Schüler, Lehrer und Eltern ist das Schulbuch zu ATARI 600/800XL vom Verständnis der Molekülbildung, exponentiellen Wachstum bis zum Pythagoras und Geschichtszahlen endlich spielerisch gut aufgerefert – viele interessante Programme. Vor allem Schüler der Mittel- und Oberstufe werden Mathe, Bio, Physik, Chemie, Sprachen und anderen Fächern Wieder fit.

DAS SCHULBUCH ZU ATARI 600/800XL, 1984, über 100 Seiten, DM 49,–



Dies sollte Ihr erstes Buch zum ATARI 600 und 800XL sein. Es ist eine sehr leichtverständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung der ATARI 600/800XL – Vorkenntnisse sind nicht erforderlich. Ergänzt wird der Text durch zahlreiche Abbildungen und Fotos. Genau das richtige Buch zum Einsteigen ins Programmieren mit ATARI 600 und 800XL.

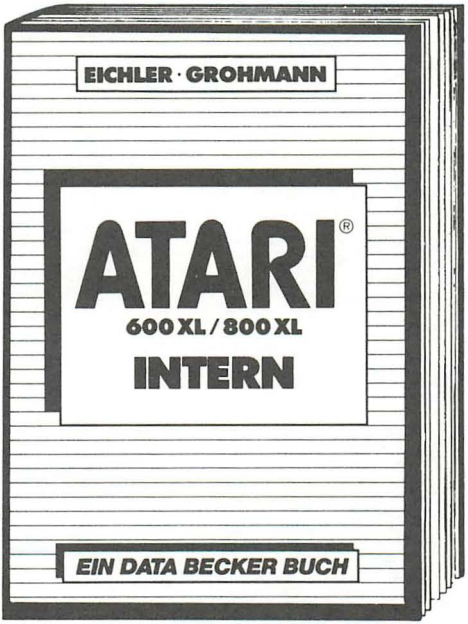
ATARI 600/800XL FÜR EINSTEIGER, 1984, über 200 Seiten, DM 29,–

Die neuen DATA BECKER BÜCHER

Dies sollte Ihr erstes Buch zum ATARI 600 und 800XL sein. Es ist eine sehr leichtverständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung der ATARI 600/800XL – Vorkenntnisse sind nicht erforderlich. Ergänzt wird der Text durch zahlreiche Abbildungen und Fotos. Genau das richtige Buch zum Einsteigen ins Programmieren mit ATARI 600 und 800XL.

ATARI 600/800XL FÜR EINSTEIGER, 1984, über 200 Seiten, DM 29,–

Die DATA BECKER GmbH

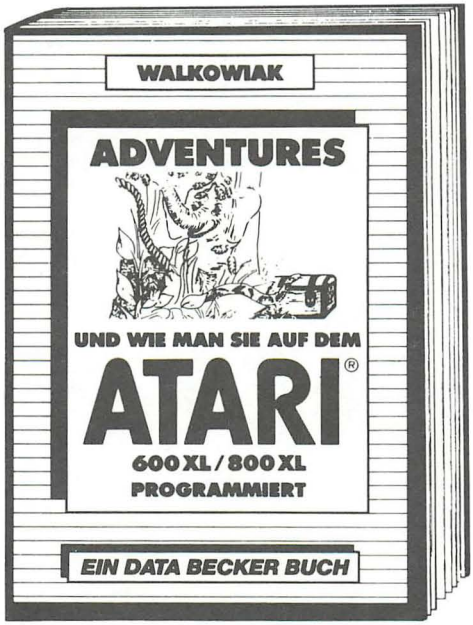


Der neue Hit aus der INTERN Serie. Beschreibt ausführlich Hardwarekonzept, ANTIC, GTIA, POKEY, PIA und das Betriebssystem der ATARI Computer sowie den Speicheraufbau. Ein unentbehrliches Hilfsmittel für jeden, der sich mit Technik und Betriebssystem der ATARI Computer 600 XL/800 XL/400/800 auseinandersetzen will. ATARI INTERN, über 300 Seiten, DM 49,–.

Die neuen DATA BECKER BÜCHER

Die neuen DATA BECKER BÜCHER

Der neue Hit aus der INTERN Serie. Besch ibt ausführlich Hardwarekonzept, ANTIC, GTIA, POKEY, PIA und das Betriebs-system der ATARI Computer sowie den Speicherausbau. Ein unentbehrliches Hilfsmittel für jeden, der sich mit Technik und Betriebssystem der ATARI computer 600 XL/800 XL/400/800 auseinandersetzen will. ATARI INTERN, über 300 Seiten, DM 49,–.

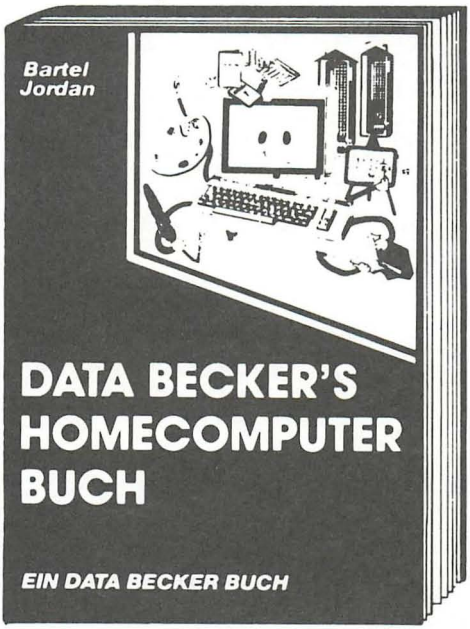


Alles über die faszinierende Welt der Abenteuerspiele. Komplett mit Adventures zum Abtippen und einem ADVENTURE-GENERATOR. ADVENTURES UND WIE MAN SIE AUF DEM ATARI 600/800 XL PROGRAMMIERT. Ca. 230 Seiten, DM 39,–

Die neuen DATA BECKER BÜCHER

Alles über die faszinierende Welt der Abenteuerspiele. Komplett mit ADVENTURES zum Abtippen und einem ADVENTURE-GENERATOR. ADVENTURES UND WIE MAN SIE AUF DEM ATARI 800/600 XL PROGRAMMIERT. ca. 220 Seiten, DM 39,–

DIE DATA EDUCATIVE



Faszinierend, was so ein Homecomputer alles kann. Dieses leicht verständliche Buch, das keinerlei Computerkenntnisse voraussetzt, hilft Ihnen nicht nur bei der richtigen Kaufentscheidung. Es berät Sie auch umfassend beim sinnvollen Einsatz Ihres eigenen Computers. Wichtige Informationen, wertvolle Ideen und nützliche Vorschläge zum Thema
HOMECOMPUTER auf über 380 Seiten für nur DM 29,–

Die neuen DATA BECKER BÜCHER

DAS STEHT DRIN:

Dieses Buch bietet eine Einführung in die faszinierende Welt der Strategiespiele. Von einfachen Spielen mit feststehender Strategie über komplexe Spiele mit komplizierten Suchverfahren bis zu lernfähigen Programmen geschieht das leicht verständlich und anhand vieler interessanter Beispiele:

- NIM mit einem Haufen
- Blockade
- Treib die Dame in die Ecke
- Springer gegen Bauer
- Vier gewinnt
- Bauernkampf
- Hexapawn
- Mini-Dame

UND GESCHRIEBEN HAT DIESES BUCH:

Walter Schneider, Student der Mathematik, ist passionierter Schachspieler und programmiert schon seit vielen Jahren auf seinem ATARI.

ISBN 3-89011-077-0